UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Alexandre Reeberg de Mello

SISTEMA DE INSPEÇÃO VISUAL DE PLACAS DE CIRCUITO IMPRESSO PARA LINHAS DE PRODUÇÃO EM PEQUENAS SÉRIES EM UM CONTEXTO MULTIAGENTES

Florianópolis (SC)
2015

Alexandre Reeberg de Mello

SISTEMA DE INSPEÇÃO VISUAL DE PLACAS DE CIRCUITO IMPRESSO PARA LINHAS DE PRODUÇÃO EM PEQUENAS SÉRIES EM UM CONTEXTO MULTIAGENTES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para obtenção do grau de "Mestre em Engenharia de Automação e Sistemas".

Orientador: Prof. Dr.-Ing. Marcelo Ricardo Stemmer, UFSC.

Florianópolis (SC)
2015

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Mello, Alexandre Reeberg de
SISTEMA DE INSPEÇÃO VISUAL DE PLACAS DE CIRCUITO IMPRESSO PARA LINHAS DE PRODUÇÃO EM PEQUENAS SÉRIES EM UM CONTEXTO MULTIAGENTES / Alexandre Reeberg de Mello ; orientador, Marcelo Ricardo Stemmer - Florianópolis, SC, 2015.
109 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Inteligência artificial. 3. Processamento de imagens. 4. Visão computacional. 5. Sistema multiagentes. I. Stemmer, Marcelo Ricardo . II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Alexandre Reeberg de Mello

SISTEMA DE INSPEÇÃO VISUAL DE PLACAS DE CIRCUITO IMPRESSO PARA LINHAS DE PRODUÇÃO EM PEQUENAS SÉRIES EM UM CONTEXTO MULTIAGENTES

Esta dissertação foi julgada aprovada para a obtenção do grau de "Mestre em Engenharia de Automação e Sistemas" e aceita em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis (SC), 9 de Junho de 2015.

Prof. Dr. Romulo Silva de Oliveira
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

---

Prof. Dr.-Ing. Marcelo Ricardo Stemmer
UFSC
Orientador

Banca Examinadora:

Prof. Dr.-Ing. Marcelo Ricardo Stemmer
UFSC
Presidente

Prof. Dr. Mario Lucio Rolloff
IFSC

Prof. Dr. Jomi Fred Hubner
UFSC

Prof. Dr. Tiago L. F. da Costa Pinto
EMC/UFSC

Este trabalho é dedicado aos colegas de profissão, amigos e familiares.

Agradecimentos

Ao orientador, Prof. Dr. -Ing. Marcelo Ricardo Stemmer, pela orientação e formação durante o período de mestrado. Aos familiares, que me suportaram e apoiaram durante este momento. À minha noiva, Jessica Freitag de Mello, pela presença e incentivo durante todos os momentos. A Deus.

O cientista não é o homem que fornece as verdadeiras respostas; é quem faz as verdadeiras perguntas.

– Claude Lévi-Strauss

Resumo

A produção em pequenas séries (PPS) vem se destacando e aumentando sua representatividade no cenário econômico, principalmente quando o assunto é tecnologia. Atualmente, percebe-se que na produção de placas de circuito impresso (PCI), há uma necessidade de produzir de maneira personalizada e eficiente, logo, há um esforço da comunidade científica e industrial em aprimorar técnicas de processamento de imagens para a inspeção de PCI.

Este trabalho inclui o estado da arte da inspeção óptica automática de PCI, não necessariamente aplicada a PPS. As técnicas utilizadas neste projeto visam a formação de um sistema para a inspeção de componentes do tipo SMD em uma PPS, garantindo uma qualidade de produção satisfatória, com a finalidade de reduzir o retrabalho. Com isso, foi proposto um sistema de inspeção baseado em características relacionadas ao contorno, posicionamento e histograma dos componentes. Este sistema é composto pelas seguintes três etapas: pré-processamento de imagens, extração de características e avaliação de componentes.

A máquina de inspeção utilizada neste projeto está inserida no contexto de cooperação entre máquinas, com o objetivo de constituir uma fábrica totalmente autônoma, coordenada por um sistema multiagente. Desta forma, foi desenvolvido um método de interação entre a máquina de inspeção e um agente que a representa.

Um software de inspeção foi implementado utilizando as arquiteturas e estruturas descritas ao longo desta dissertação. Os resultados obtidos pelos experimentos mostram-se adequados à inspeção de componentes SMD em uma PPS, pois apontam uma taxa de acerto acima de 89% ao utilizar componentes reais.

Palavras-chave: Inspeção de PCI, processamento de imagens, visão computacional, inspeção óptica automática, inteligência artificial, agentes.

Abstract

Small series production (SSP) has been higlighting and increasing its share in the economic scenario, especially when it comes to technology. Currently, it is seen that in printed circuit boards (PCB) production, there is a need to produce in a customized and efficient way, so an effort of the scientific and industrial community is present, to improve the image processing techniques for PCB inspection.

It is present in this work the state of the art for automatic optical inspection in a PCB, not necessarily applied to SSP. The techniques used in this project aim the formation of a system to inspect SMD components in a SSP, ensuring a satisfactory production quality, with the purpose of reduce the rework. Therefore, an inspection system based on characteristics related to shape, positioning and histogram of components has been proposed. This system is compound of the following three steps: pre-processing of images, feature extraction and evaluation components.

The inspection machine used in this project is inserted in the context of cooperation among machines in order to provide a fully autonomous factory, coordinated by a multi-agent system. Thus, a interaction method between the inspection machine and a agent was developed.

An inspection software was implemented using the architecture and structures described throughout this dissertation. The results obtained by experiments show that it is suitable for inspection of SMD components in a SSP, showing a success rate above 89% when using actual components.

Keywords: PCB inspection, image processing, computer vision, automatic optical inspection, aritificial intelligence, multiagent systems.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Sumário

Capítulo 1

INTRODUÇÃO

A produção de placas de circuito impresso (PCI) está em crescimento de forma expressiva e constante nos contextos econômico e tecnológico. Economicamente, este é evidenciado através da participação de US$ 60 bilhões no mercado mundial [IPC, 2014], valores apenas da produção de manufatura em eletrônicos e montagem de equipamento final. Tecnologicamente é evidenciado pelo aumento de funcionalidades nos equipamentos eletrônicos, conjunto com a redução das dimensões destes mesmos equipamentos, acarretando em uma maior densidade de integração de semicondutores e interconexões nas PCIs, desta forma, um aumento na quantidade de componentes por placa produzida. Não apenas se tratando da quantidade de componentes, há também uma evolução tecnológica do próprio componente, como melhor integridade de sinal e maiores frequências de *clock*.

Concomitantemente com a evolução tecnológica veio a personalização de equipamentos eletrônicos, e trouxe consigo o aumento da quantidade de produção em pequenas séries (PPS). Este tipo de manufatura adota várias formas e perspectivas, e diferentemente da produção em larga escala, onde se tem geralmente um único produto a ser fabricado, existe uma variedade de produtos à serem produzidos em um curto espaço de tempo [HITOMI, 1996], ou seja, exige um processo de manufatura flexível e de alta confiabilidade, para que resulte em produtos de qualidade. Cada item produzido por um lote de pequena série possui um alto valor agregado em relação à produção total, logo é indesejado qualquer descarte de produto devido à falha de fabricação. Para minimizar os eventuais problemas durante a produção, é comumente adotado o uso de inspeção de qualidade durante a produção, com o intuito de identificar o quanto antes qualquer eventualidade. Na produção de PCI, essas inspeções devem ser capazes de avaliar a qualidade

de trilhas, defeitos de placa ou componentes, e qualidade de instalação. Atualmente existem vários métodos para realizar a inspeção de PCI, os quais são categorizados em com ou sem contato físico. No contexto de inspeção sem contato, encontra-se a inspeção óptica automática, capaz de reconhecer defeitos de maneira não destrutiva e precisa.

Com o aprimoramento de técnicas e equipamentos, a inspeção visual torna-se uma prática cada vez mais confiável e frequente, substituindo a tradicional, e propensa a falhas, inspeção humana. As técnicas e aplicações envolvendo a área de visão computacional tem evoluído rapidamente, e esta evolução acontece devido ao barateamento e aperfeiçoamento de câmeras e dispositivos de processamento, assim como o desenvolvimento de algoritmos mais robustos e sofisticados utilizados no campo da visão computacional [KAEHLER; BRADSKI, 2015].

1.1. OBJETIVOS DO TRABALHO

1.1.1. Objetivo geral

A finalidade deste trabalho é a pesquisa e desenvolvimento de um método de inspeção visual automática de placas de circuito impresso através de técnicas de processamento digital de imagens, a fim de garantir a qualidade de uma PCI em uma produção em pequena série, assim como o desenvolvimento de um protocolo de comunicação entre o sistema de inspeção proposto e um agente que o represente. À vista disso, este trabalho possui seu enfoque na inspeção de componentes, sem qualquer tipo de marcação, do tipo SMD (*Surface Mounted Devices*) já montados em placas.

1.1.2. Objetivos específicos

Os seguintes objetivos específicos foram planejados para atingir o objetivo geral:

- elaborar um método de processamento de imagens que possibilite a inspeção automática adequada para componentes do tipo SMD sem qualquer tipo de marcação no contexto de uma PPS;
- implementar um software de inspeção baseado no método desenvolvido;
- elaborar um protocolo de comunicação entre a máquina de inspeção S2iAOI e um agente que represente este software;
- validar o software e o sistema de comunicação implementados, utilizando fotos de componentes reais do tipo SMD já instalados

em uma PCI, adquiridas com sistema de visão computacional S2iAOI, desenvolvido no projeto BRAGECRIM 013/09.

1.2. ORGANIZAÇÃO DA DISSERTAÇÃO

O capítulo 2 localiza o trabalho presente no contexto do projeto BRAGECRIM 013/09, responsável pela motivação inicial deste e demais projetos.

A fundamentação teórica utilizada para o desenvolvimento deste trabalho é vista no capítulo 3.

O sistema de inspeção visual desenvolvido e o protocolo de comunicação entre o agente e a máquina S2iAOI, incluindo a estrutura do software, as arquitetura de processamento de imagem e classificação, a validação dos métodos de classificação, e o sistema de comunicação entre agente e máquina são apresentadas no capítulo 4.

No capítulo 5 é apresentada a experimentação do software de inspeção proposto.

O capítulo 6 expõe as conclusões desta dissertação, assim como sugestões para desenvolvimentos futuros, visando a melhora e expansão do sistema de inspeção e do protocolo de comunicação agente-máquina desenvolvido neste trabalho.

Capítulo 2

CONTEXTUALIZAÇÃO DO TRABALHO

A produção em pequenas séries (PPS), ou em pequenos lotes (PPL), é a produção em que uma grande variedade de produtos é manufaturada em um curto espaço de tempo, resultando em um baixo volume de produção [HITOMI, 1996]. Desta forma, este tipo de produção é fundada na flexibilidade, pois atende a crescente demanda de produtos personalizados produzidos em lotes reduzidos em um curto espaço de tempo, que surgem com rápida evolução de determinados nichos do mercado, principalmente o tecnológico. Desta forma não há um único modo de produção, contudo características em comum de PPS são encontradas em [JADHAV, 2005], [DORO, 2009] e [SCHMITT; PAVIM, 2009], e corroboram com a característica de flexibilidade de produção, como as seguintes evidenciadas por [SZYMANSKI, 2014]:

- alta variedade de produtos a serem produzidos;
- curto tempo de produção por lote;
- baixo volume de produção, com a possibilidade de produzir um único item.

A produção em corrida curta (PCC) está fortemente associada à PPS, pois é a manufatura de produtos com um curto ciclo de vida, demandado por uma variedade de clientes, fazendo que o ambiente de produção tenda a produzir uma maior diversidade de produtos com lotes cada vez menores, [LIN et al., 1997]. Segundo [JADHAV, 2005], o baixo volume ou pequeno lote são os termos mais utilizados para definir a PCC, os mesmos encontrados na PPS, além de destacar que os setups frequentes são características do PCC, problema abordado por este trabalho.

As estratégias de medição utilizadas para verificar a qualidade de produções em larga escala são dependentes de características fixas de

algumas variantes da manufatura em questão, além da possibilidade de controlar ou compensar essas variantes ao longo da produção [SCHMITT; PAVIM, 2009]. PPS necessitam uma outra abordagem, pois existe a necessidade de garantir um certo nível de qualidade já no primeiro lote, e este acaba por se tornar o desafio da PPS segundo [SCHMITT; PAVIM, 2009]. O mesmo autor ainda destaca que o pouco tempo existente para observar e corrigir o processo durante a produção, falta de previsibilidade sobre o processo e seu comportamento, dificuldade em reutilizar informações para tomar ações corretivas e o elevado tempo de setup inicial aliado a poucos produtos descartáveis (ou nenhum) a serem usados para ajustar os processos são obstáculos referentes a garantir a qualidade em uma PPS.

Com o intuito de melhorar a performance existente, foi criado o projeto de metrologia cognitiva para a produção flexível em pequenas séries, ou COGMET (*Cognitive Metrology for Flexible Small Series Production*), com uma solução inovativa para aumentar a eficiência de produção dentro de linhas de produções flexíveis, ao desenvolver uma nova geração de sistema metrológicos e de garantia da qualidade da produção com capacidades adaptativas e alto grau de percepção cognitiva em relação ao produto e ao processo [PFEIFER et al., 2010] [SZYMANSKI, 2014]. Este projeto se encontra inserido no programa colaborativo brasileiro alemão de iniciativa à pesquisa de tecnologia de manufatura, ou BRAGECRIM (*Brazilian German Collaborative Research Initiative on Manufacturing Technology*), formado pelo grupos de pesquisa S2i/DAS/UFSC e o LABelectron/Certi, apoiado pela CAPES, FINEP e CNPq, na parte brasileira, e o instituto WZL/RWTH-Aachen, apoiado pelo DFG, na parte alemã. O COGMET utiliza o conceito de sistema auto-organizável [FRANK et al., 2004], que é caracterizado pela capacidade de analisar a situação corrente, determinar novos objetivos de sistema, assim como adaptar o comportamento do sistema para novas condições de ambiente [PFEIFER et al., 2010]. A figura 2.1 ilustra a definição de um sistema auto-organizável.

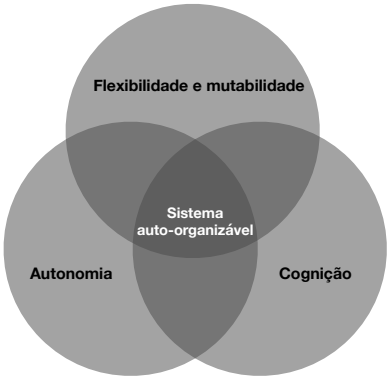


Figura 2.1: Definição de um sistema auto-organizável, adaptado de [STEMMER et al., 2014]

Desta forma os aspectos mais relevantes são [SCHMITT; PAVIM, 2009]:

- flexibilidade e mutabilidade: perceber as diferenças externas estimuladas pelo ambiente e se adaptar à situações novas ou pré-definidas;
- autonomia: reagir pró-ativamente contra mudanças do ambiente e guiar o sistema para estados desejados e seguros de operação, sem interferência de operadores externos;
- cognição: aprender pelas próprias experiências, tomar decisões inteligentes e agir de volta ao ambiente em uma maneira segura e robusta.

Dentro do contexto do projeto COGMET, o diagnóstico de falhas em uma produção foi investigado, e um sistema especialista (SE)[1] foi proposto. O sistema de comunicação entre o SE e a planta é feito através de um sistema multiagente [WEISS, 1999] [WOOLRIDGE; WOOLDRIDGE, 2001] (que engloba todas as máquinas de manufatura), proposto por [ROLOFF, 2009], e aborda a configuração e monitoramento de uma PPS. Este age ao definir o modelo de referência, conceber a arquitetura de referência para configuração e o monitoramento da PPS, e apresentar uma arquitetura de implementação genérica baseada no modelo de referência e na arquitetura de referência. Desta forma o SE se torna um agente de diagnóstico, que procura atingir os objetos previamente designados [COSTA, 2012]. A topologia do agente de

[1] técnica de IA composta por uma máquina de inferência e uma base de conhecimento, construída a partir da expertise de um ou mais especialistas.

diagnóstico é representado pela figura 2.2.

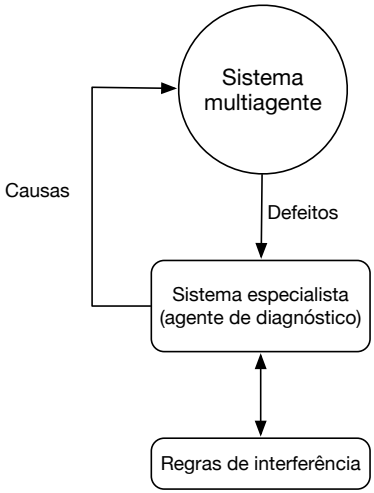


Figura 2.2: Topologia geral de um sistema especialista, adaptado de [STEMMER et al., 2014]

O experimento simulado utilizado neste projeto empregou como cenário o processo de montagem de PCI para uma PPS, que inclui as tecnologias THT (*Through-Hole Technology*) e SMT (*Surface Mounted Technology*), cujo estudo relacionado às falhas de produção de PCI, e suas possíveis causas, foi feito por [DORO, 2009]. No LABelectron, laboratório-fábrica pertencente à fundação CERTI, PCIs são montadas utilizando ambas tecnologias, em lotes de 30 unidades. A figura 2.3 representa o fluxo de montagem utilizando a tecnologia SMT, que é o foco deste trabalho.

O processo se inicia na máquina *Printer*, que aplica a pasta de solda responsável por fixar os componentes SMD às placas. A SPI (*Solder Paste Inspection*) tem o papel de inspecionar se a aplicação da solda foi feita de maneira correta. As máquinas de inserção, tanto a automática quanto a manual, são responsáveis por posicionar os componentes na placa e o forno de refusão solda efetivamente estes componentes. Ao fim da linha de montagem há uma máquina de inspeção ótica automática de componentes (*Automatic Optical Inspection*, AOI) utilizada para averiguar a qualidade da produção. Como a inspeção está posicionada após à solda dos componentes, as placas com problemas são

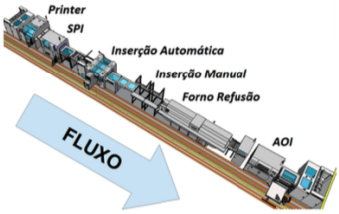


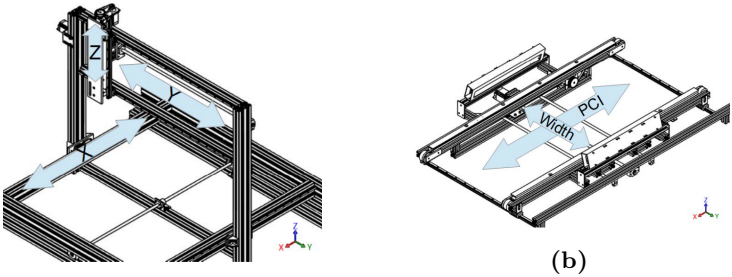
Figura 2.3: Linha de montagem SMT do LABelectron. Fonte [MELO, 2013].

descartadas, pois o custo do retrabalho é geralmente inviável quando os componentes estão fixados na placa.

Com a possibilidade em inspecionar qualquer etapa da linha de montagem SMT, uma máquina com característica flexível de AOI foi desenvolvida por [MELO, 2013]. Esta possibilita a instalação de novos sistemas de iluminação e aquisição de imagem, assim como novas técnicas de inspeção, devido à sua arquitetura aberta. Além da flexibilidade já descrita, a máquina provê possibilidade de movimentação no chão de fábrica, integração com outras máquinas, ajuste automático da largura do *conveyor* (transportador) e possibilidade de agentificação. O protótipo da máquina de AOI é denominado S2iAOI (*Automated Optical Inspection machine for Industrial Intelligent Systems group*), e é apresentado pela figura 2.4.

Figura 2.4: Sistema de visão computacional S2iAOI. Fonte [SZYMANSKI, 2014].

A câmera instalada na máquina S2iAOI se movimenta em três graus de liberdade sobre a PCI no espaço cartesiano, como é representado pela figura 2.5a, e o ajuste da largura do *conveyor* é representado pela figura 2.5b.



(a) **(b)**

Figura 2.5: Representação gráfica do sistema de movimentação da câmera (a) e ajuste de largura do *conveyor* (b).

O software de interface gráfica principal do protótipo, também desenvolvido por [MELO, 2013], possibilita o usuário criar novos programas de inspeção, efetuar uma calibração personalizada para cada projeto, e encontrar o ponto zero da máquina através de duas marcas fiduciais. Um primeiro método de inspeção óptica automática foi realizada por [SZYMANSKI, 2014], cujo foco foi inspecionar PCI baseadas em componentes com algum tipo de marcação de identificação de superfície ou serigrafia. O método utiliza o algoritmo de SIFT (*Scale Invariant Feature Transform* - Transformação de descritores invariantes a escala), desenvolvido por [LOWE, 1999] e atualizado por [LOWE, 2004], para extrair pontos chaves, ou *keypoints*, que representem características desejadas dos objetos. Estes descritores são invariantes a escala, translação e rotação de imagens, e parcialmente invariante a mudança de iluminação ou projeção, desta forma é capaz de identificar os seguintes defeitos: componente ausente, rotacionado/invertido, deslocado e errado. Os keypoints são obtidos através da identificação de pontos estáveis (ou pontos de interesse) formados a partir de uma filtragem no espaço de escala. Dados do histograma das vizinhanças de cada ponto obtido, geram a chave SIFT, onde ao fim do processo um vetor de chaves para cada imagem é formado. Desta forma, a comparação entre objetos ou pontos é feita através da relação de semelhança entre estas chaves, como é exemplificado pela figura 2.6.

Figura 2.6: Resultado da correspondência entre fotos de componentes utilizando SIFT. Fonte [SZYMANSKI, 2014].

Os *keypoints* gerados em componentes sem marcação são instáveis, pois são formados a partir da variação de padrões existentes da imagem, neste caso a variação de contraste. Devido a existência de imperfeições na superfície do componente, os *keypoints* formados refletem estas imperfeições, e não características do componente, tornando a técnica imprópria para componentes sem qualquer tipo de marcação. A arquitetura representada pela figura 2.7 expõe o fluxo do software de inspeção definido por [SZYMANSKI, 2014].

Dentro do contexto apresentado, o trabalho presente é composto de dois objetivos:

- desenvolver um software para inspeção óptica automatizada que age de maneira complementar ao proposto por [SZYMANSKI, 2014], isto é, realiza a inspeção da qualidade da instalação de componentes do tipo SMD em PCIs, os quais não possuam qualquer tipo de marcação ou serigrafia (a figura 2.8b ilustra a diferença entre os tipos de componente referente a marcação ou serigrafia). Este processo de inspeção é composto de métodos de processamento e segmentação de imagem, representação de objeto por descritores e técnicas de inteligência artificial para reconhecimento de objetos.
- possibilitar a comunicação entre os softwares de interface e inspeção existentes com a produção coordenada pelo sistema multiagente, assim como criar um agente que represente a máquina de inspeção.

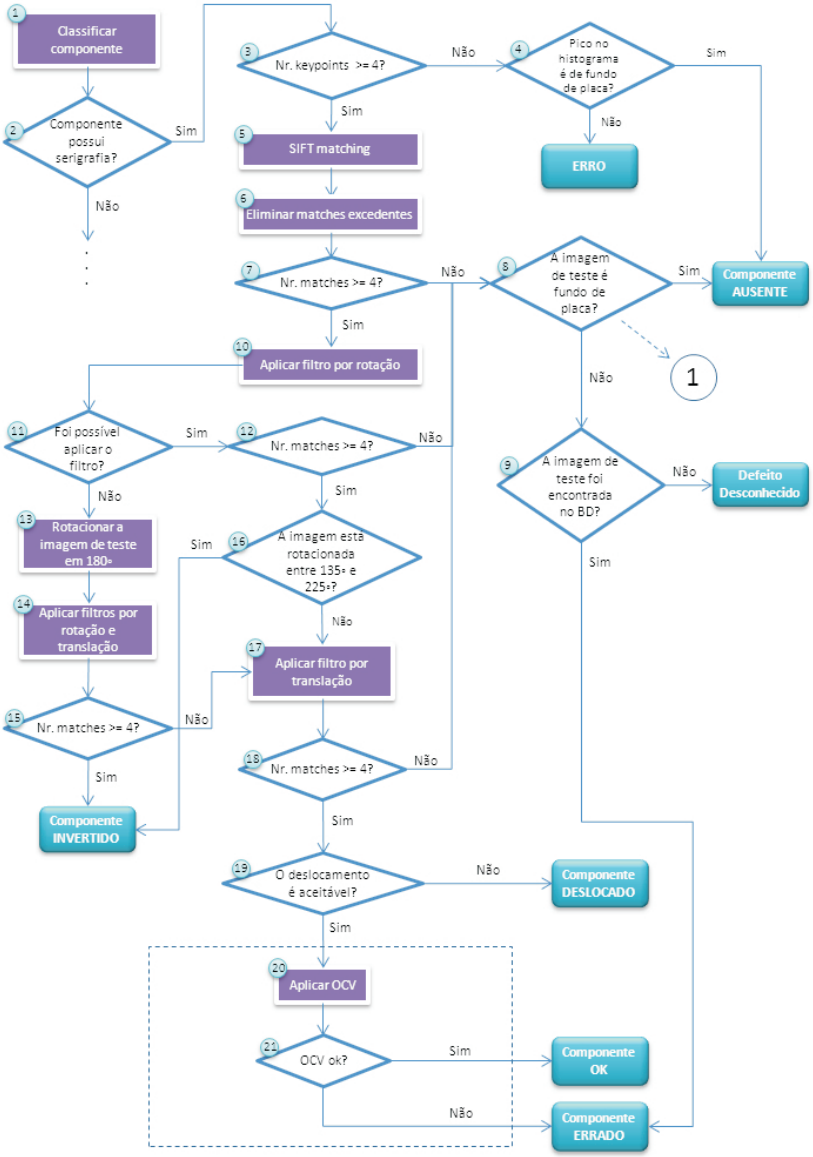
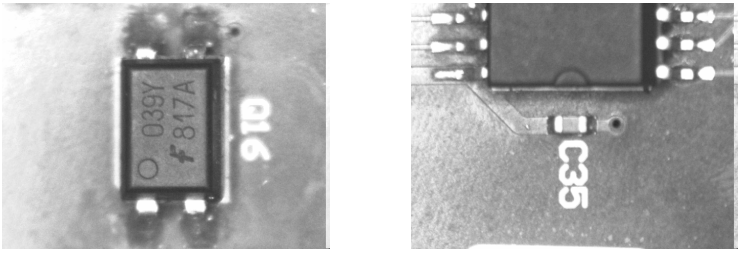


Figura 2.7: Fluxograma da arquitetura de processamento de imagens proposta. Fonte [SZYMANSKI, 2014].



(a) **(b)**

Figura 2.8: Exemplo de componentes eletrônicos com (a) e sem (b) serigrafia.

Capítulo 3

FUNDAMENTAÇÃO TEÓRICA

Os principais conceitos necessários para compreender este trabalho são apresentados neste capítulo, assim como uma contextualização. Primeiramente serão expostos os trabalhos relacionados, seguido da qualidade da produção em pequenas séries, e técnicas de inspeção de PCI. As técnicas de processamento de imagens serão então detalhadas, assim como a representação de imagens via descritores. Os métodos utilizados de reconhecimento de objetos através de IA são fundamentados, assim como uma forma de validação e a agentificação de sistema.

3.1. TÉCNICAS DE INSPEÇÃO DE PLACAS DE CIRCUITO IMPRESSO

Métodos de detecção de falhas em PCI podem ser divididos em duas amplas categorias: métodos com contato (ou elétrico) e métodos sem contato (não elétricos).

Segundo [MOGANTI et al., 1996], métodos com contato são utilizados apenas para encontrar erros de curto circuito ou circuito aberto, e não são capazes de acusar erros cosméticos ou erros que possam vir a acontecer ao longo do uso do equipamento, como uma redução indevida da espessura do trilho. Um outro porém é o custo necessário para se inspecionar uma placa utilizando métodos elétricos, pois quando se envolve uma engenharia de precisão, os acessórios necessários para se avaliar uma PCI são caros e complexos.

Os métodos sem contato são baseados em diferentes tecnologias, e os mais utilizados são:

- inspeção por raio-X;
- laminografia;
- inspeção acústica;

- inspeção térmica;
- inspeção automática visual.

Sistemas de inspeção por raio-X são constituídos de uma fonte de emissão de raios que produzem um área de radiação e um detector de imagem, com isso as imagens de raio-X são obtidas a partir do escaneamento da PCI. É utilizada para inspecionar rapidamente e com precisão PCIs com multicamadas [MOGANTI et al., 1996], pois se destaca pela possibilidade de detectar erros sob a superfície [O'CONCHUIR et al., 1991], desta forma, é capaz de identificar erros como pequenas rachaduras, problemas relacionados à solda ou componentes desalinhados.

Laminografia é utilizada para separar as camadas da PCI em diferentes imagens utilizado a inspeção por raio-X. Esse processo é feito movendo-se a fonte de emissão de raio-X e o detector de imagem em volta da PCI, mas em direções opostas. Com isso os raios atravessam, ao mesmo tempo, os mesmos pontos na PCI e no detector, criando uma imagem transversal [MOGANTI et al., 1996] [O'CONCHUIR et al., 1991].

Inspeção acústica é realizada através da análise de imagens adquiridos por sistemas sônicos ou ultrassônicos, na qual a diferença entre esses dois sistemas é a frequência da onda de som utilizada, cujo quaisquer valor de frequência acima de 400Khz é caracterizado de ultrassônico [O'CONCHUIR et al., 1991]. Ambos sistemas de emissão de som usam transdutores piezoelétricos para gerar pulsos focalizados de energia alta frequência. Ao monitorar a reflexão ou transmissão desses pulsos, defeitos relacionados à solda, juntas ou fissuras na placa podem ser identificados [SAVAGE et al., 1993].

A inspeção térmica possibilita duas diferentes abordagens de inspeção: imagem térmica e inspeção de perfil térmico [O'CONCHUIR et al., 1991]. A imagem térmica é obtida através da diferença de níveis de radiação infravermelha emitida por componentes e soldas entre uma PCI ligada e desligada. A inspeção de perfil térmico é feita através do sistema de inspeção à laser Vanzetti, no qual um determinado componente ou junta é avaliado ao longo de tempo em relação a sua variação de temperatura quando exposto ao determinado laser. A vantagem desse sistema é a possibilidade de ser aplicado quando a PCI está em funcionamento [MOGANTI et al., 1996].

O sistema de inspeção automática visual (*Automatic Visual Inspection*, AVI) é realizado através da análise de imagens obtidas por câmeras, desta forma, visa a identificação de falhas de superfície, ou seja, não é possível identificar defeitos internos ou escondidos [SAVAGE et al., 1993]. Técnicas de processamento de imagem e classificação são

comumente utilizadas para realizar inspeção ótica, e desenvolvidas para resolver problemas críticos da indústria de manufatura que envolvam engenharia de precisão [HUANG; PAN, 2015]. Os sistemas atuais de inspeção podem ser categorizados em quatro grupos:

- métodos de projeção: modelam a correlação entre exemplos aprendidos e uma determinada característica que representam esses exemplos [HUANG; PAN, 2015]. Os métodos mais utilizados são a análise de componente principal (*Principal component analysis*, PCA), a análise independente de componente (*Independent component analysis*, ICA) e a análise discriminante linear (*Linear discriminant analysis*, LDA).
- abordagem baseada em filtro: métodos de filtros espaciais são utilizados para melhorar a resolução e qualidade de sistemas óticos, com isso são habitualmente utilizados em aplicações de melhoramento de imagem ou técnicas de restauração de imagens [HUANG; PAN, 2015]. As técnicas mais utilizadas são os filtros passa-baixa, passa-alta, assim como transformações de sinal para domínio de frequência, como a transformada de Fourier, Wavelet ou discreta de cosseno.
- abordagem baseada em aprendizagem: são desenvolvidos com aprendizado de máquina e algoritmos de reconhecimento de padrão. Os algoritmos de aprendizagem mais utilizados são redes neurais artificiais, algoritmos genéticos e máquinas de vetores de suporte [HUANG; PAN, 2015].
- métodos híbridos: são sistemas que combinam várias técnicas de inspeção para realizar tarefas complexas.

3.1.1. Defeitos usuais em PCI

Defeitos de placas de circuito impresso são principalmente causados por falta ou excesso de elementos na placa. Estes defeitos podem ser categorizados em dois grupos: funcionais ou cosméticos [WU et al., 1996]. Defeitos funcionais podem afetar a performance da PCI assim como causar sua falha, enquanto defeitos cosméticos podem prejudicar a performance da placa ao longo do tempo devido a anormalidades na dissipação de calor ou na distribuição de corrente [Indera Putera; IBRAHIM, 2010] [HAZURAH et al., 2012]. Os defeitos podem ser relacionados à placa ou aos componentes inseridos nesta placa, e estes apresentam diferentes características a serem analisadas, desta forma o método de inspeção deve ser orientado à necessidade.

O rompimento ou falha na impressão de alguma trilha, curto cir-

cuito ou a falta de um furo, no caso de placas que utilizam componentes THT (*Through-Hole Technology*) são os defeitos relacionados à placa mais procurados [HAZURAH et al., 2012] [Indera Putera; IBRAHIM, 2010] [LIN et al., 1997] [MOGANTI et al., 1996].

Após a inserção de componentes na placa deve-se garantir que os mesmos estejam corretamente posicionados e inseridos. Para isso o sistema deve inspecionar se há algum componente ausente, rotacionado/invertido, deslocado em relação aos *pads* ou errado (inserção de um componente indevido no lugar de outro) [WU et al., 2010], [LETA et al., 2008], [Han-Jin Cho; Tae-Hyung Park, 2008].

Existem ainda defeitos que não são relacionados à montagem, mas ao transporte, manuseio e armazenamento de componentes e placas, bem como a má qualidade dos mesmos [DORO, 2009] [SZYMANSKI, 2014].

3.2. TRABALHOS RELACIONADOS

Interpretar imagens utilizando um conjunto de técnicas e métodos através de sistemas computacionais é o papel da visão computacional. Essa interpretação é a transformação de um conjunto de dados que representam uma imagem tridimensional em uma estrutura de dados, como projeções bidimensionais [JAIN et al., 1995]. O desenvolvimento de novas técnicas de visão computacional e áreas relacionadas, como aquisição e processamento de imagens e inteligência artificial, cria soluções eficientes para problemas atuais, como inspecionar a qualidade de PCI durante produção.

O trabalho de [Han-Jin Cho; Tae-Hyung Park, 2008] propõe um método de inspeção de componentes do tipo SMD por meio de um algoritmo de template matching projetado para reduzir o tempo de cálculo, assim como a capacidade de armazenamento de imagem, como é ilustrado na figura 3.1.

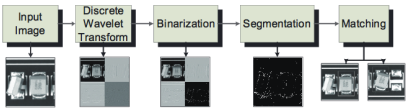


Figura 3.1: Fluxo do método de template matching. Fonte [Han-Jin Cho; Tae-Hyung Park, 2008]

O autor transforma todas as imagens de entrada e referência do domínio espacial em um domínio comprimido, utilizando o método de

transformada discreta de Wavelet (DWT, *discrete wavelet transform*) nas orientações horizontal e vertical, obtendo imagens em escala de cinza. Em seguida estas imagens são binarizadas e segmentadas, eliminando pixels que estão localizados fora da borda do objeto. Por fim é executado o algoritmo de template matching, o qual se baseia na relação entre as funções de diferença média absoluta (MAD, *mean absolute difference*) e a diferença média dos quadrados (MSD, *mean square difference*), onde ambos calculam a diferença entre o brilho do pixel da imagem de referência e da imagem de entrada. Em um experimento com 30 imagens diferentes de tamanho $256x256$, a taxa de compressão de imagem foi de 0.071, reduzindo o tamanho de memória inicial de 192KB para 13.67KB para cada imagem. O tempo de cálculo médio foi de 44ms, e utilizando os métodos tradicionais de MAD ou MSE em imagens sem compressão o resultado foi de 897ms. A precisão do sistema de inspeção proposto é ligeiramente (varia de 1% a 5% dependendo do problema analisado) menos eficiente quando comparado aos métodos tradicionais já citados.

O sistema proposto por [WANG et al., 2009] realiza a inspeção da serigrafia contida nos componentes de uma PCI já carregada, cujos métodos utilizados são o *Mean Shift*, proposto por [FUKUNAGA; HOSTETLER, 1975] e desenvolvido por [CHENG, 1995], e o algoritmo rápido de *Supported Vector Machine* (SVM), desenvolvido [PLATT, 1998]. Como a marcação nos componentes é feita com letras de forma, não existem grandes variações, pois apenas a fonte utilizada para marcar o componente pode mudar, tornando o SVM possível para classificação de componentes. O processo se inicia com a aplicação do método de mean-shift, seguido de um algoritmo adaptativo de segmentação, com a finalidade de extrair uma imagem binária com a região textual do componente, onde a partir desta imagem se realiza a segmentação do texto por caracteres, formando por sua vez um vetor de características geradas pela projeção da serigrafia, como é exemplificado pela figura 3.2.

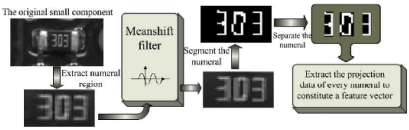


Figura 3.2: Ilustração da extração de características. Fonte [WANG et al., 2009]

O sistema de classificação baseado em SVM ocorre em dois pas-

sos: treinamento de exemplos e classificação de dados desconhecidos. Os exemplos são divididos nas categorias positivo e negativo, onde a primeira contém números que devem ser inspecionados, e a segunda os valores não desejados. Desta forma o resultado do treinamento é um hiperplano que divide os bons dos maus exemplos, e a classificação de um componente a ser testado produz o resultado de correto ou errado. Em um experimento realizado foram utilizados 60 exemplos positivos e 200 negativos do numeral "0", e 65 positivos e 220 negativos do numeral "473"para o treinamento, e inspecionados duas PCI com 271 numerais, onde haviam 38 numerais "0"e 33 "473". No fim do experimento houveram 5 erros de classificação.

O estudo de [LUO et al., 2013] visa a inspeção de componentes SMD propondo um método de *image matching*, que engloba a correção de posição e verificação de caracteres. Para isso um modelo de referência é formado por imagens de componentes corretamente instalados, e uma análise de posição e ângulo é feita a fim de estipular qual a máxima variação aceita. Para se realizar a inspeção, primeiramente é feito um cálculo de correlação entre o componente a ser analisado e o componente de referência, onde caso o resultado do cálculo esteja dentro do limite pré-estabelecido, o componente analisado é digitalmente corrigido, se necessário, em relação a ângulo e posição. Para a inspeção de caracteres, o componente em teste é segmentado através de um algoritmo dinâmico que utiliza limiares em escala cinza, determinado pelo histograma, proposto por [Bo Peng; Tianrui Li, 2013], resultando em uma imagem binarizada que contém apenas os dígitos, como é mostrado na figura 3.3. Os caracteres do componente analisado são separados individualmente,

Figura 3.3: Segmentação e imagens negativas de componentes. Fonte [LUO et al., 2013]

e uma análise conduzida por uma rede neural artificial verifica se estes são compatíveis com o modelo respectivo. A rede proposta é constituída de uma camada intermediária composta de dois neurônios, cuja função de ativação é a sigmoidal. Toda imagem de caracteres é normalizada para o tamanho de 5 linhas por 8 colunas, e o treinamento desta rede é feito com caracteres de várias fontes utilizando diferentes tipos de componentes. 2180 componentes com diferentes tipos de serigrafia foram testados, e a taxa de falso positivo obtida foi de 0.0917% (ou apenas 2

componentes), contra 1,42% (ou 31 componentes falsamente detectados) por um método que utiliza apenas a rede neural artificial, sem a correção de posicionamento.

A estratégia proposta por [LIN; SU, 2006] é uma inspeção rápida de PCI realizada em duas etapas, onde a primeira é baseada na comparação de índices e a segunda é uma rede neural artificial que utiliza método de propagação retroativa. Os índices utilizados pelo autor são:

- quantidade de pixels brancos: abordagem proposta por [TEOH et al., 1990], é utilizado para detectar erro na orientação do componente, onde o índice utilizado é obtido através da razão entre o número de pixels brancos e a quantidade total de pixels da imagem;
- histograma: na proposta desenvolvida por [TEOH et al., 1990], a diferença na distribuição dos níveis de cinza é utilizada para detectar a falta ou desalinhamento de componente. O índice é obtido pela relação entre a diferença da distribuição dos níveis de cinza e a quantidade total de pixels da imagem;
- coeficiente de correlação: introduzido por [PERNG et al., 2011], a correlação de coeficiente detecta componentes ausentes, e o coeficiente é o determinado pela média, variância da imagem testada e uma imagem de referência;
- valor regional: índice utilizado para reduzir o efeito da iluminação, é similar ao índice de histograma, contudo os níveis de cinza são divididos em menos níveis;
- valor de contraste alto: o índice é obtido pela contagem de pixels com valor relativamente alto resultante da subtração entre a imagem examinada e a imagem de referência respectiva.

Dentre os índices apresentados, o que obteve a melhor resposta, em uma primeira etapa onde não se inclui a falta de componente, foi o índice de coeficiente de correlação, com uma taxa de falso positivo de 36.5%, seguido de 43% do índice de valor de contrasto alto, e 70% ou mais para os demais. Os componentes que falharam à comparação de índices são re-avaliados na segunda etapa. Foram testadas três configurações de rede neural utilizando os índices como entrada, RNA3-7-2, RNA4-9-2 e RNA5-9-2, onde todas utilizaram o mesmo conjunto de treinamento, formado por 558 imagens, obtidas por variações, digitalmente criadas, de 18 componentes. Um teste com 1949 exemplos coletados, em que 29 diferentes defeitos estavam presentes, foi realizado utilizando as 3 configurações propostas. A rede RNA3-7-2 obteve uma taxa de falta de componentes (*missing rate*) de 58,54% e 0% da classificação equivocada. A RNA4-9-2 obteve uma taxa de falta de componente e

de componentes classificados erroneamente de 14,63%, enquanto a rede RNA5-9-2 obteve uma taxa de falta de componente de 2,44% e 4,88% de erro de classificação equivocada.

Uma aplicação de inspeção óptica e classificação de juntas de solda, feitas para a tecnologia SMT, utilizando rede neural é proposta por [ACCIANI et al., 2006]. O procedimento de inspeção é formado pela aquisição de imagem, pré-processamento, extração de características e classificação, como é apresentado pela figura 3.4 . A etapa de pré-

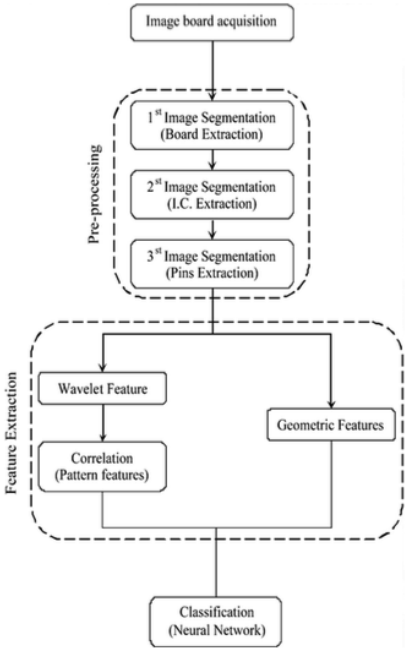


Figura 3.4: Procedimento de inspeção. Fonte [ACCIANI et al., 2006]

processamento consiste em obter a região de interesse da imagem. O processo da segmentação se inicia extraindo a borda sob inspeção da imagem adquirida, seguido da obtenção da imagem do componente, e por fim extraindo as juntas do componente a ser testado. Dois tipos de características da imagem da solda foram utilizadas pelo autor, a geométrica e a *wavelet*. O vetor de características geométricas é formado pela área, perímetro, compacidade, altura, largura e comprimento, enquanto o vetor de características wavelet é formado pelos 6 primeiros índices da transformada discreta de wavelet. Um terceiro vetor de

características é formado pela união dos dois vetores anteriormente citados. Desta forma, três tipos de classificadores foram utilizados, uma rede neural LQV (*Learning Vector Quantization*), uma rede neural MLP (*Multilayer Perceptron*) e o KNN (*K Nearest Neighbor*). Uma validação cruzada de "n" pastas foi implementada para se obter a arquitetura das redes neurais, comparando o resultado obtido entre várias arquiteturas testadas. Três testes com 240 imagens foram realizados, onde o primeiro utiliza apenas características geométricas, o segundo apenas as wavelets e o último um híbrido geométrico-wavelet. A eficiência da rede neural MLP no primeiro teste foi de 97,5%, seguido de 94,6% da LQV e 94,74% do KNN. Para o segundo teste, a MLP resultou em 95,8%, a LQV obteve 95,4% e o KNN 93,75%. No último teste, a MLP teve sucesso em 98,8%, a LQV 97,1% e o KNN 95,79%.

3.3. REVISÃO DE TÉCNICAS DE PROCESSAMENTO DE IMAGEM DE INTERESSE

A partir do estudo de trabalhos relacionados, foram identificados algumas técnicas de interesse para o problema a ser tratado nesta dissertação.

Processamento de imagens gráficas realizadas por computador refere-se a informações geométricas de objetos, ou modelos geométricos, que são expressas pela descrição matemática, ou conceito, representadas através de um computador para mostrar, armazenar, modificar e melhorar a operação e os processos associados à imagem [LV; WU, 2011]. Uma imagem digital pode ser definida como uma função, limitada, de quantidades discretas e de duas dimensões, $f(x, y)$, no qual x e y são coordenadas espaciais de um plano, e a amplitude de f para qualquer par de coordenadas (x, y), é chamada de intensidade, ou nível de cinza, para esta determinada imagem. [GONZALEZ; WOODS, 2006]. A quantidade finita de elementos existentes em uma imagem digital que definem a função possuem valores e posições particulares, e são denominados pixels [GONZALEZ; WOODS, 2006]. A partir desta definição é possível manipular a imagem através de operações realizadas nos pixels, resultando em transformação, melhoramento, segmentação ou reconhecimento de imagem.

3.3.1. Histograma

Histograma de uma imagem monocromática é a representação gráfica da frequência de ocorrência de cada nível de cinza da imagem

[MARQUES, 2011]. Uma imagem com pouca variação do nível de cinza, por exemplo, a superfície de uma PCI sem componentes, o qual apenas a cor do fundo, trilhos e alguma marcação pode ser encontrada, apresenta aglomerações dos níveis de cinza em determinadas regiões do gráfico de histograma, mantendo um padrão de comportamento. Ao adicionar um componente à esta PCI, o padrão do histograma muda, pois o componente possui diferentes tons de cinza. Desta forma, comparar padrões de histograma é o método utilizado neste projeto para verificar a existência de um componente.

A quantidade de níveis de cinza de uma imagem digital é definida pela variação $[0, L-1]$, e o histograma é definido pela função discreta $h(r_k) = n_k$, onde r_k é o k-ésimo valor de intensidade e n_k é o número de pixels de uma imagem de intensidade r_k [GONZALEZ; WOODS, 2006]. Os histogramas são normalmente normalizados e apresentados em forma de gráfico de barra, no qual o eixo x se refere aos valores de níveis de cinza e o eixo y ao valor proporcional (ou percentual) de pixels equivalente ao particular valor. O histograma normalizado é definido por $p(r_k) = \frac{n(k)}{MN}$, para $k = 0, 1, 2, ..., L-1$ e uma imagem de tamanho MxN. Logo $p(r_k)$ é a estimativa da probabilidade de ocorrência da intensidade de nível r_k de uma imagem [GONZALEZ; WOODS, 2006] [MARQUES, 2011].

3.3.2. Processamento de imagem morfológica

Morfologia matemática provê uma abordagem de processamento de imagens baseado no formato, nos quais seus operadores tendem a simplificar as informações da imagem preservando suas características essenciais de formato e eliminando irrelevâncias [HARALICK et al., 1987]. Uma variedade de operadores são obtidos quando considerados como conjuntos teóricos operacionais em imagens binarizadas [FORSYTH; PONCE, 2002], e os membros deste conjunto, que representam as imagens binarizadas, de um espaço inteiro 2-D, Z^2, em que cada elemento de um conjunto é uma tupla (vetor 2-D), são coordenadas (x, y) de pixel branco, ou preto dependendo da convenção utilizada [GONZALEZ; WOODS, 2006]. Conjuntos euclidianos 3-D podem conotar uma variação da imagem binária ao longo do tempo ou uma imagem estática em escala de cinza, e ainda podem haver diferentes perspectivas para os demais N-espaços euclidianos [HARALICK et al., 1987]. Os operadores apresentados a seguir, dilatação e erosão, assim como seus derivados abertura e fechamento, são utilizados para facilitar o isolamento da área de interesse, isto é, remover ruídos indesejados das imagens dos

componentes.

3.3.2.1. Dilatação e erosão

As transformações morfológicas de dilatação e erosão são realizadas através da adição ou subtração de subconjuntos de elementos. O subconjunto que sofre a transformação morfológica, é usualmente chamado de elemento de estrutura, e os elementos utilizados geralmente são agrupados, isto é, não se utiliza um único pixel por elemento, mas sim uma vizinhança de kxk [FORSYTH; PONCE, 2002] [GONZALEZ; WOODS, 2006].

Dilatação é a transformação morfológica que combina dois conjuntos através da adição vetorial de subconjuntos de elementos [HARALICK et al., 1987], e é usualmente utilizado para agrupar ou engrossar linhas em uma imagem binarizada. A definição é dada por, quando A e B são subconjuntos de um espaço N-dimensional (E^N), com elementos $a = (a_1, ..., a_n)$ e $b = (b_1, ..., b_n)$, sendo os elementos uma N-tupla de coordenadas, então a soma é dada por [HARALICK et al., 1987]:

$$A \oplus B = \{c \in E^N | c = a + b, \quad \text{para algum} \quad a \in A \quad \text{e} \quad b \in B\} \quad (3.1)$$

Erosão é o dual morfológico da dilatação, logo é a transformação morfológica que combina dois conjuntos usando subtração de subconjuntos [HARALICK et al., 1987]. A erosão tem como propósito encolher ou afinar objetos em uma imagem binária, com isso pode ser considerada como uma operação morfológica de filtragem, nos quais os detalhes presentes na imagem que são menores que a estrutura de elemento são removidos [GONZALEZ; WOODS, 2006] Considerando o mesmo espaço N-dimensional utilizado na transformação de dilatação, a erosão de A em B é o conjunto de todos elementos x, tal que $x + b \in A$ para todo $b \in B$ [HARALICK et al., 1987], desta forma, a erosão é definida como:

$$A \ominus B = \{x \in E^N | x + b \in A, \quad \text{para todo} \quad b \in B\} \quad (3.2)$$

3.3.2.2. Abertura e fechamento

Dilatação e erosão são geralmente utilizadas em pares, no qual se aplica uma dilatação no resultado de uma erosão ou vice-versa. Estas operações são idempotentes, isto é, os efeitos de reaplicações não modificam os resultados de transformações prévias [HARALICK et al., 1987]. A abertura é utilizada para suavizar contornos, quebrar istmos estreitos e eliminar pequenas ilhas ou pontos agudos [HARALICK et al., 1987].

Para uma imagem B submetida ao processo de abertura, utilizando a estrutura de elemento K, sua definição é dada por [HARALICK et al., 1987][GONZALEZ; WOODS, 2006]:

$$B \circ K = (B \ominus K) \oplus K \tag{3.3}$$

O fechamento é utilizado para suavizar o contorno, fundir estreitas lacunas e golfos longos e finos, eliminar pequenos buracos e preencher falhas de contorno [HARALICK et al., 1987]. Para uma imagem B submetida ao processo de fechando, utilizando a estrutura de elemento K, sua definição é dada por [HARALICK et al., 1987][GONZALEZ; WOODS, 2006]:

$$B \bullet K = (B \oplus K) \ominus K \tag{3.4}$$

3.3.3. Segmentação de imagem

O termo *segmentação de imagem*, segundo [STOCKMAN; SHAPIRO, 2001], refere-se a uma parte de imagem que se encontra dentro do conjunto de regiões que cobre esta determinada parte. Para [RAUT et al., 2009], é a técnica que particiona uma imagem de entrada em regiões únicas de pré-requisitos semântico. A segmentação tem como objetivo decompor a imagem, no caso o componente a ser analisado, para análise futura ou realizar uma mudança na representação [STOCKMAN; SHAPIRO, 2001]. As técnicas mais comuns utilizadas na área de segmentação de imagens são [LALA; JAIN, 2013]:

- baseada em limiar: normalmente aplicada em imagens em nível de cinza, a divisão por limiar é realizada através da comparação de valor dos pixels com um limiar escolhido, dividindo a imagem em duas partes, nas quais uma corresponde ao fundo (*background*) e a outra aos objetos de interesse (*foreground*) [LALA; JAIN, 2013]. O algoritmo proposto por [OTSU, 1979] é uma junção de técnicas baseada em limiar e histograma.
- baseada em histograma: diferentes agrupamentos são formados a partir do histograma de uma imagem, geralmente expressos por picos e vales. Com isso métodos de busca em agrupamentos e classificação por padrão são utilizados para formar grupos de acordo com a necessidade, como os propostos por [DUDA et al., 2000]. Desta forma, as técnicas baseadas em histogramas são utilizadas como refinamento para os processos posteriores.
- detecção de borda: uma borda é um conjunto de conexões, i.e., de mesmo nível de intensidade, entre dois pixels adjacentes e que possam ser distintos pela estimativa da intensidade do gradiente

[NETRAVALI; HASKELL, 1988]. As bordas em uma imagem podem ser dividas em duas categorias, bordas de intensidade e bordas de textura. Bordas de intensidade surgem a partir de mudanças abruptas no perfil de intensidade de uma imagem [TAN et al., 1989]. Bordas de textura são fronteiras de regiões de textura, as quais são invariantes à iluminação [LALA; JAIN, 2013]. A maioria das técnicas abordam a detecção de ambas categorias de borda, além de utilizar diferentes procedimentos para inserir os pixels de fronteira em bordas com significado. Os algoritmos mais famosos são: detecção de borda proposta por Sobel(1968), detecção proposta por Prewitt(1970), detecção utilizando aproximação de derivativas proposta Roberts (1965) e detecção através da procura de máxima local do gradiente utilizando a derivativa do filtro gaussiano proposta por Canny(1986) [CANNY, 1986].

- baseada em região: é dividido em dois grupos, região de crescimento e divisão e fusão de região. O procedimento de crescimento, agrupa pixels ou sub-regiões em maiores regiões baseadas em um critério pré-definido [LALA; JAIN, 2013]. O procedimento de divisão e fusão é realizado através da subdivisão aleatória de uma imagem em regiões disjuntas, e em seguida realizada a fusão ou separação para satisfazer as restrições de pré-requisito, logo o algoritmo é realizado de maneira iterativa [LALA; JAIN, 2013].
- transformação *watershed*: A transformação de watershed considera a magnitude do gradiente de uma imagem como uma superfície topográfica. Pixels com o maior valor de intensidade da magnitude de gradiente (*gradient magnitude intensity*, GMIs) correspondem às linhas de watershed, as quais representam regiões de fronteiras. Água posicionada em qualquer pixel anexo à uma linha comum de watershed vaza em declive em direção a uma mínima de intensidade local (*local intensity minima*, LMI). Pixels que drenam para uma mínima em comum formam uma bacia de captação, que representam a região de interesse [GONZALEZ; WOODS, 2006].
- partição por grafos: uma imagem é segmentada e modelada como um grafo não-direcionado com pesos. Cada pixel corresponde a um nó do grafo, então a borda é formada entre cada par de pixels. O peso de uma borda é a medida da similaridade entre pixels. A imagem é particionada em conjuntos disjuntos através da remoção da conexão de bordas dos segmentos. O particionamento ótimo do grafo acontece quando o peso das bordas removidas é o mínimo [LALA; JAIN, 2013].

3.3.3.1. Método Otsu

O método de segmentação de imagem baseado na seleção automática de limiar de forma não-paramétrica e não-supervisionada foi proposto por [OTSU, 1979]. O método é considerado ótimo, pois além de maximizar a variância entre classes, ele é baseado apenas em cálculos computacionais utilizando o histograma 1-D da imagem [GONZALEZ; WOODS, 2006]. Para uma imagem composta de apenas um objeto e um fundo o histograma é bimodal, onde cada um dos componentes é representado por uma moda, facilitando a escolha dos limiares.

Sendo uma imagem digital, em escala de cinza, de tamanho MxN pixels, e L a intensidade dos níveis desses pixels, que no caso variam de $\{0, 1, 2, ..., L-1\}$, e n_i denotando o número de pixels com a intensidade i, o total de pixels em uma imagem é $MN = n_0+n_1+n_2+...+n_{L-1}$. O histograma normalizado possui componentes $p_i = n_i/MN$, resultando em:

$$\sum_{i=0}^{L-1} p_i = 1, \quad p_i \geq 0 \tag{3.5}$$

Ao aplicar um determinado limiar $T(K) = k,\ 0 < k < L-1$ em uma imagem, esta é separada em duas classes, C_1 e C_2, cujos valores de intensidade de C_1 variam de $[0, k]$ e C_2 os valores variam de $[k+1, L-1]$. Desta forma, a probabilidade de um pixel pertencer à classe C_1 é dada por:

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{3.6}$$

Similarmente, a probabilidade da classe C_2 é:

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k) \tag{3.7}$$

A média dos valores de intensidade dos pixels pertencentes à classe C_1 é:

$$m_1(k) = \frac{1}{P_1(k)} \sum_{i=0}^{k} ip_i \tag{3.8}$$

Similarmente, a média dos valores de intensidade de C_2 é:

$$m_2(k) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} ip_i \tag{3.9}$$

A média cumulativa até o nível k é:

$$m(k) = \sum_{i=0}^{k} ip_i \tag{3.10}$$

Desta forma, a média de intensidade da imagem inteira é:

$$m_G = \sum_{i=0}^{L-1} ip_i \tag{3.11}$$

A fim de avaliar a efetividade do algoritmo no nível k, a métrica, sem dimensão, normalizada η é utilizada

$$\eta = \frac{\delta_B^2}{\delta_G^2} \tag{3.12}$$

onde δ_G^2 é a variância global, e δ_B^2 a variância entre classes, cuja última, quando inserido o valor de k, é expressa por:

$$\delta_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \tag{3.13}$$

Com isso, o valor ótimo do limiar k é obtido quando a variância entre classes é máxima:

$$\delta_B^2(k^*) = \max_{0 \leq k \leq L-1} \delta_B^2(k) \tag{3.14}$$

Assim que k^* é obtido, a imagem é segmentada por:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > k^* \\ 0 & \text{if } f(x,y) \leq k^* \end{cases} \tag{3.15}$$

para $x = 0, 1, 2..., M-1$ e $y = 0, 1, 2, ..., N-1$. A métrica normalizada η, calculada para o valor ótimo de limiar, $\eta(k^*)$ pode ser utilizada para obter a estimativa quantizada da separabilidade de classes, com valores entre $0 \leq \eta(k^*) \leq 1$. O limite inferior é atingido apenas por imagens únicas com nível de intensidade constante, enquanto o limite superior é atingido por imagens de 2 valores com intensidades entre 0 e $L-1$ [GONZALEZ; WOODS, 2006].

3.3.3.2. Segmentação usando morfologia de bacia hidrográfica (*Watershed*)

A segmentação de Watershed é baseada em região, diferentemente dos métodos apresentados anteriormente, os quais são baseados em

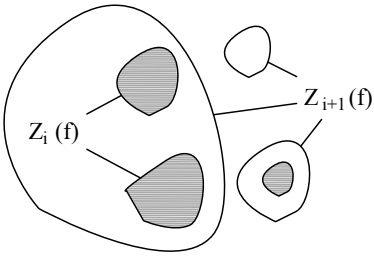


Figura 3.5: Construção de bacia hidrográfica utilizando o esqueto por zona de influência (SKIZ) geodésico, adaptado de [BEUCHER; MEYER, 1993]

contorno. Neste tipo de segmentação, o contorno também é obtido ao final do algoritmo, contudo é uma consequência, pois em algum momento do método há a detecção de todo o objeto de interesse. Com isso é possível identificar se o objeto de interesse foi erroneamente dividido, ou ainda verificar se existe mais de um objeto na imagem.

O conceito de bacia hidrográfica é baseado na visualização da imagem em três dimensões: duas coordenadas espaciais versos a intensidade, interpretando esta imagem como uma uma forma topográfica [BEUCHER; MEYER, 1993] [BEUCHER; LANTUEJOUL, 1979]. Três tipos de pontos necessitam ser considerados: (a) pontos que estão em um mínimo local, (b) pontos que se comportam como uma gota d'água, se posicionados em uma região de (a), irão cair para um único mínimo, e (c) pontos podem estar posicionados em um situação em que caem para mais de um mínimo local. Os pontos que satisfazem a condição (b) são chamados de ponto de bacia-hidrográfica ou poço da região, e os que satisfazem (c) estão localizados em linhas de divisão, ou linhas de bacia-hidrográfica [GONZALEZ; WOODS, 2006].

Segundo [BEUCHER; MEYER, 1993], considerando que a inundação atinga o nível máximo expresso pela seção $Z_i(f)$ da figura 3.5, fica claro que a inundação de $Z_{i+1}(f)$ acontece nas zonas de influência de componentes conectados por $Z_i(f)$, desta forma os componentes não afetados pertencentes a $Z_{i+1}(f)$ são por definição, mínimos no nível $i+1$.

Essas mínimas devem ser adicionadas à área de inundação. Denotando $W_i(f)$ a seção no nível i do poço, e $m_{i+1}(f)$ a mínima da função

no peso $i+1$, a seção é definida por:

$$W_{i+1}(f) = [IZ_{Zi+1(f)}(X_i(f)) \cup m_{i+1}(f)] \tag{3.16}$$

e a mínima é definida por:

$$m_{i+1}(f) = Z_{i+1}(f)/RZi + 1(f)(Z_i(f)) \tag{3.17}$$

A iteração do algoritmo se inicia com $W_{-1}(f) = \emptyset$, e ao final do processo, as linhas da bacia hidrográfica $DL(f)$ são iguais a:

$$DL(f) = W_N^c(f) \quad \text{com } max(f) = N \tag{3.18}$$

Ao fim do processo de iteração, todas as mínimas regionais se tornam centros de poços, logo uma super-segmentação é formada na imagem. Entretanto nem todas as mínimas regionais são de mesma importância, pois algumas representam ruídos ou estruturas não desejadas da imagem. Para resolver este problema, o conceito de marcador é utilizado, o qual consiste em um componente conectado à imagem [GONZALEZ; WOODS, 2006]. Existem dois tipos de marcadores, o interno, que é associado à componentes de interesse, e o externo, associado ao fundo da imagem (*background*). Os marcadores internos estão localizados próximos a pontos de maiores altitudes, de forma que o conjunto de pontos nessa região formem um componente conectado, e todos os componentes conectados devem ter o mesmo valor de intensidade.

O algoritmo de Watershed deve ser aplicado em uma imagem suavizada, então a imagem é pré-processada através das operações morfológicas de abertura, fórmula 3.3, seguida de um fechamento, fórmula 3.4. Com isso, o algoritmo é aplicado com a restrição de apenas os marcadores internos serem mínimas regionais permitidas, limitando a região de atuação do algoritmo. As linhas resultantes do algoritmo de bacia hidrográfica formam os marcadores externos, que são regiões compostas de um único marcador interno e uma parte do fundo da imagem.

Ao adicionar a seleção de marcadores, a definição do algoritmo é agora dada por:

$$W_{i+1}(g) = IZZi + 1 \cup M(W_i(g)) \tag{3.19}$$

onde $W_i(g)$ é a seção no nível i de um novo poço g, com

$$W_{-1}(g) = M, \quad \text{conjunto de marcadores} \tag{3.20}$$

Para uma melhor seleção de marcadores, a função gradiente g é modificada para um novo gradiente g', formando uma imagem

muito semelhante à original mas com um conjunto de marcadores M que substitui as mínimas iniciais. Esta modificação é conhecida por homotopia. Com isso

$$\forall i, \quad Z_i(g') = R_{Z_i(g) \cup M}(M) \tag{3.21}$$

e ao denotar por k_m a função indicadores dos marcadores, e por i_{max} o máximo valor de g,

$$k' = i_{max}(1 - k_m) \tag{3.22}$$

e então

$$g' = R^*_{Inf(g,k')}(k') \tag{3.23}$$

Desta forma o algoritmo de Watershed pode ser aplicado diretamente sobre o gradiente modificado g', não necessitando a mínima real de g, reduzindo o custo computacional.

3.3.3.3. Detecção de bordas Canny

O operador de detecção de borda, utilizado neste projeto, foi proposto por [CANNY, 1986], e possui uma performance satisfatória quando os parâmetros utilizados no método são ajustados para um cenário específico, como é o caso deste projeto, cuja iluminação é constante e os componentes possuem características semelhantes. Canny possui melhor desempenho neste caso, quando comparado, por exemplo, com os métodos de [ROTHWELL et al., 1995], [BERGHOLM, 1987], [IVERSON; ZUCKER, 1995] e [NALWA; BINFORD, 1986], como conclui [HEATH et al., 1997]. O operado possui três principais características [NIXON; AGUADO, 2008]:

- detecção ótima de bordas sem respostas falsas: Canny reduz a resposta à ruídos através do uso do filtro Gaussiano.
- boa localização com distância mínima entre a posição da borda detectada e a verdadeira: detecta picos através da supressão não-máxima, retendo apenas os pontos no topo do cume da informação da borda.
- resposta única para eliminar múltiplas respostas de uma única borda: localiza um ponto único de borda que é responsável pela mudança de brilho.

Para satisfazer as características do algoritmo, 4 passos são necessários [NIXON; AGUADO, 2008]:

- suavizar a imagem: a suavização é realizada através de um filtro gaussiano.
- computar os gradientes: computar a magnitude dos gradientes e os ângulos das imagens.

- supressão não-máxima: aplicar a supressão não-máxima na magnitude do gradiente da imagem.
- duplo limiar: são utilizados dois limiares e análise de conectividade para detectar e juntar bordas.

Para o primeiro passo do algoritmo [SZELISKI, 2010], é necessário realizar a suavização da imagem utilizando a diferenciação do operador gaussiano $G_\sigma(x) = \exp(\frac{-(x^2+y^2)}{2\sigma^2})$, desta forma, a função núcleo Gaussiano é:

$$\nabla G_\sigma(x) = (\frac{\partial G_\sigma}{\partial x}, \frac{\partial G_\sigma}{\partial y})(x) = [-x, -y]\frac{1}{\sigma^3}\exp(-\frac{x^2+y^2}{2\sigma^2}) \quad (3.24)$$

onde o parâmetro σ indica a largura do Gaussiano.

Desta maneira, a suavização é à convolução do núcleo Gaussiano com a imagem em que se deseja suavizar, logo o gradiente dessa operação pode ser escrito como:

$$J_\sigma(x) = \nabla[G_\sigma(x) * I(x)] = [\nabla G_\sigma](x) * I(x) \quad (3.25)$$

O próximo passo é calcular a máxima (magnitude do gradiente) em uma borda, cuja direção é perpendicular à orientação desta mesma borda, seguindo a direção do gradiente. Para isso, é calculado a convolução do gradiente 3.25 com o Laplaciano do Gaussiano (LoG):

$$S_\sigma(x) = \nabla \cdot J_\sigma(x) = [\nabla^2 G_\sigma](x) * I(x) \quad (3.26)$$

onde o cerne do LoG é definido por:

$$\nabla^2 G_\sigma(x) = \frac{1}{\sigma^3}(1-\frac{x^2}{2\sigma^2})G_\sigma(x)G_\sigma(y)+\frac{1}{\sigma^3}(1-\frac{y^2}{2\sigma^2})G_\sigma(y)G_\sigma(x) \quad (3.27)$$

Nesta etapa é necessário identificar os elementos que não possuem cruzamento, chamados de *zero crossing*, e convertê-los em elementos de borda (*edgels*). Para isso, procura-se os pixels adjacentes x_i e x_j em que os sinais mudam de valor, i.e., $[S(x_i) > 0] \neq [S(x_j) > 0]$. A localização do sub-pixel é obtida conectando os valores de $S(x_i)$ e $S(x_j)$,

$$x_z = \frac{x_i S(x_j) - x_j S(x_i)}{S(x_j) - S(x_i)} \quad (3.28)$$

Desta forma a orientação e força dos edgels podem ser obtidos através da interpolação linear dos valores dos gradientes na grade dos pixels originais.

Ao identificar todos os edgels, é necessários juntá-los em uma lista encadeada, começando com o primeiro edgel encontrado e adicionando

os sucessivos adjacentes. A representação desta lista mais utilizada é feita através da parametrização do comprimento de arco, $x(s)$, no qual s denota o comprimento do arco ao longo da curva. Por fim o limiar com histerese é aplicado, o qual necessita dois diferentes limiares. O processo identifica um edgel que possui valor acima do limiar superior estabelecido, e identifica seus vizinhos. Qualquer vizinho com valor acima do limiar inferior é considerado um ponto da borda, e o processo segue ao longo de todos os edgels. Caso os pontos excedam o limiar inferior, ficam marcados como um ponto *seed*, e um novo ramo de busca é feito a partir destes pontos. Para cada ramo, a busca termina no ponto em que não se encontra vizinhos acima do limite inferior [NIXON; AGUADO, 2008] [SZELISKI, 2010].

3.3.3.4. Algoritmo seguidor de borda

O algoritmo seguidor de borda proposto por [SUZUKI; BE, 1985] tem como propriedade extrair a estrutura topológica de uma imagem binária, criando marcas únicas para cada borda ao invés de utilizar uma mesma marca para todas as bordas, e adicionando um procedimento para se obter as bordas parentes da borda atualmente analisada [SUZUKI; BE, 1985]. Neste projeto, será utilizado o algoritmo seguidor de borda considerando apenas as bordas mais externas, entre o componente e o fundo de tela, também proposto por [SUZUKI; BE, 1985].

Para melhor compreender o algoritmo, algumas considerações e definições são necessárias. Um ponto de borda é definido como um pixel de valor 1, (i, j), com toda a vizinhança de valor 0, (p, q). Uma borda externa é definida como um conjunto de pontos de borda entre um componente de valor 1, escolhido arbitrariamente, em que componentes com valor 0 o cercam diretamente, desta forma para um componente arbitrário com valor 1, a borda externa é única.

O algoritmo inicia varrendo a imagem, de cima para baixo e da esquerda para direita, até encontrar um ponto de borda, o qual um valor único é atribuído, chamado de número sequencial de borda (NSB). A partir deste momento, o algoritmo varre a imagem marcando os pixels pertencentes a esta borda, identificado-os com o valor de NSB da borda. Se a borda atualmente analisada estiver entre um componente de valor 0 que contém um pixel $(p, q + 1)$ e um componente que contém um pixel (p, q), o valor do pixel é modificado para -NSB. Caso contrário, o valor do é modificado para NSB, a não ser que este já pertence a alguma borda. Ao finalizar o processo, uma nova varredura da imagem se inicia, afim de identificar novas bordas. Sempre que uma nova borda é identificada, se esta é mais externa que à anterior, assume o valor de

$NSB + 1$, caso contrário é descartada. A borda parente mais externa à borda do objeto é o fundo da tela, desta forma o ponto da borda (i, j) imediatamente à direita de um pixel com valor 0, é a borda mais externa do objeto apenas se satisfazer uma das seguintes condições:

- todos os pixels $(i, 1), (i, 2), ..., (i, j - 1)$ possuem valor 0.
- o último ponto de borda, (i, h), encontrado faz parte da borda mais externa, e o pixel $(i, h + 1)$ faz parte do fundo de tela.

Como os pixels do fundo de tela possuem valor 0, e o algoritmo apenas analisa a borda mais externa do objeto em questão, o maior valor possível de NSB é 2, pois a borda com valor 1 representa o objeto. Desta forma o algoritmo não identifica furos ou componentes dentro de componentes.

3.4. RECONHECIMENTO DE OBJETOS

Um sistema de reconhecimento de objetos tem a capacidade de achar objetos no mundo real através de uma imagem, utilizando modelos de objeto conhecidos previamente [JAIN et al., 1995]. Um tema central para o reconhecimento de objetos é o conceito de aprendizagem de padrões, que são regiões individuais de uma imagem que caracterizam um objeto. As técnicas de reconhecimento baseadas em aprendizagem utilizam descritores quantitativos (teórico-decisivo), como área, tamanho e textura, e qualitativos, como os descritores de Fourier [GONZALEZ; WOODS, 2006], como parâmetros do sistema de reconhecimento.

3.4.1. REPRESENTAÇÃO DE FORMAS E DESCRITORES

A representação de formas e técnicas de descrição podem ser classificadas nas seguintes classes de métodos: bordas ou regiões e espectros [ZHANG; LU, 2004]. Descritores espectrais representam quantidades que possam ser medidas, como intensidade de luz, gradientes locais, características estatísticas e momentos, porém envolvem maior custo computacional, onde frequentemente requerem cálculos com ponto flutuante [KRIG, 2014]. Dentre os métodos mais conhecidos encontram-se o SIFT (apresentado no capítulo anterior) e o SURF (*Speeded Up Robust Features* - Descritores robustos acelerados). Este último, proposto por [BAY et al., 2008], é baseado no SIFT, e tem como principais diferenças:

- convoluções são obtidas mais rapidamente através do uso de imagens integrais, resultando em um algoritmo mais rápido;

- detecção de pontos de interesse por meio do uso de matriz Hessiana [MIKOLAJCZYK; SCHMID, 2001];
- descritores criados a partir do uso da distribuição de respostas *Haar-wavelet*, horizontal e vertical, na vizinhança que circunda o ponto de interesse.

Os descritores baseados em bordas e regiões são divididas em estrutural (discreto) e global (contínuo), sendo que este último extrai um vetor de características multi-dimensional a partir de informações da borda de uma forma [ZHANG; LU, 2004]. Um dos métodos de representar a borda de um objeto é através da assinatura de uma forma, que representa o formato de um objeto através de uma função unidimensional derivada dos pontos da fronteira deste mesmo formato [ZHANG; LU, 2004] [Dengsheng Zhang; Guojun Lu, 2001]. Para uma representação mais robusta, isto é, com menor propensão a erros devido a ruídos ou pequenas variações de borda, as transformações espectrais são comumente aplicadas, e as técnicas mais utilizadas são os descritores de onda (*Wavelet descriptors*) e os descritores de Fourier (*Fourier descriptor*, FD).

3.4.1.1. Descritores de Fourier

Os descritores de Fourier, [ZAHN; ROSKIES, 1972], são obtidos através da transformada de Fourier aplicada na assinatura de forma da borda de um objeto, cujos coeficientes resultantes da transformada são chamados de FD do formato, e representam a forma do objeto em um domínio de frequência [Dengsheng Zhang; Guojun Lu, 2001] [GRANLUND, 1972]. Os descritores de baixa frequência contém informação sobre características gerais do objeto, enquanto os de alta frequência contém informações referente aos detalhes.

Para aplicar a transformada de Fourier é necessário normalizar a amostragem obtida, que tem como consequência a eliminação de pequenos detalhes e ruídos, além de extrair as características da linha mais externa do objeto mantendo os pontos de saliência. A normalização é obtida através da relação do número total de pontos candidatos amostrados ao longo da borda, K, e o número total de pontos pertencentes à borda, L, resultando no espaço entre dois pontos candidatos consecutivos, L/K.

A representação por distância dos pontos ao centróide, (x_c, y_c), é comumente utilizada, pois é invariante à translação. A função que define esta distância é dada por:

$$r_i = ([x_i - x_c]^2 + [y_i - y_c]^2)^{1/2}, \quad i = 1, 2, ..., L \tag{3.29}$$

onde a borda de um objeto é formada por um conjunto de coordenadas $(x_i, y;_i), i = 1, 2, ..., L$.

Para diminuir o custo computacional, a transformada rápida de Fourier (FFT) é utilizada, e quando aplicada em r_i, com $Npontos$ para $i = 1, 2, ..., N-1$, é definida por [GONZALEZ; WOODS, 2006]:

$$u_i = \frac{1}{N} \sum_{i=0}^{N-1} r_i \exp(\frac{-j2\pi ni}{N}), \quad n = 0, 1, ..., N-1 \tag{3.30}$$

e os coeficientes resultantes u_n são os FD, denotados por FD_n. Os descritores adquiridos são invariantes à translação, e para que também sejam invariantes à rotação, apenas as informações de magnitude são utilizadas. Ao dividir as magnitudes pelo componente DC, por exemplo $|FD_0|$, a invariância à escala é obtida. Por fim o vetor que indexa o formato do objeto é definido por:

$$\mathbf{f} = [\frac{|FD_1|}{|FD_0|}, \frac{|FD_2|}{|FD_0|}, ..., \frac{|FD_N|}{|FD_0|}] \tag{3.31}$$

3.4.2. MÉTODOS DE IA PARA RECONHECIMENTO DE OBJETOS

Um sistema de aprendizagem é um programa de computador que toma decisões baseadas no acúmulo de experiências contidas em casos de sucesso resolvidos, e o aprendizado consiste em escolher ou adaptar parâmetros de um modelo estrutural que melhor trabalha com os exemplos já adquiridos [WEISS; KULIKOWSKI, 1991]. Um sistema de aprendizado de alto nível baseado em tomadas de decisões é chamado de classificador [WEISS; KULIKOWSKI, 1991]. A ideia de classificação, ou predição, consiste em reconhecer objetos baseados em suas características, e o reconhecimento de padrões é um sistema de classificação [JAIN et al., 1995]. Dentro das técnicas existentes de classificação baseadas em IA, utiliza-se neste projeto o classificador por k vizinhos mais próximos (*K-Nearest Neighbor*, KNN), algoritmos de aprendizagem baseado em instância (*Instance Based Learning Algorithm*, IBL), e a técnica de rede neural, pois são utilizadas características das imagens como instâncias, além de não haver a necessidade em nenhuma das técnicas de utilizar exemplos negativos para treinamento, o que condiz com a aplicação em questão.

3.4.2.1. KNN baseado em instâncias

O método de KNN, [COVER; HART, 1967], é completamente não paramétrico, e não há forma estabelecida para definir a fronteira entre as K classes, pois o método pode produzir qualquer superfície complexa arbitrariamente para separar estas classes, baseada na configuração de pontos exemplos e seus parâmetros, ou ainda a relação de distância entre um e outro [WEISS; KULIKOWSKI, 1991]. Entre as medidas de distância para representar a proximidade mais utilizadas se encontram as distâncias absoluta, euclidiana e normalizada variada. O princípio do algoritmo de KNN é comparar o padrão de uma determinada imagem com um número de paradigmas e então classificar esta imagem de acordo com a classe de paradigmas que possuem correspondência mais próxima, contudo muitas instâncias são necessárias. Desta forma, [HART, 1968] propôs uma abordagem condensada com instâncias selecionadas, isto é, reduzindo a quantidade de informação necessária para a previsão. [AHA et al., 1991] propõe um algoritmo baseado em instâncias, utilizando o princípio de [HART, 1968], que pode ser aplicado utilizando um conceito de similaridade, classificação e descrição do conceito ou ainda através do método de classificador KNN baseado em instâncias (IBK).

O aprendizado do KNN pode ser feito com atributos de classe qualitativa (valores discretos) ou atributos de classe quantitativa (valores reais) [BATISTA; SILVA, 2009]. Neste trabalho o método de aprendizagem utilizado é o preguiçoso (*Lazy learning*), que consiste primeiramente em protelar o processamento até que este receba um pedido de informação. Em seguida a informação é respondida com a combinação da informação de treinamento. Por fim a resposta dada e resultados intermediários são destruídos [AHA, 1997]. Consequentemente a função objetivo será localmente próxima à resposta desejada do algoritmo de KNN [VIJAYARANI1; MUTHULAKSHMI, 2013]. A aprendizagem pode ser feita com valores de natureza discreta, com isso o conjunto de classes V é finita $\{v_1, ..., v_n\}$ e representado pela seguinte função $f : \Re^n -> V$. O parâmetro geralmente utilizado pelo IBK é a distância Euclidiana definida por

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{3.32}$$

onde $a_r(x)$ denota o valor do atributo r da instância x, e as previsões para mais de um vizinho (ou classe) contém um peso que é calculado de acordo com a sua distância para com a instância teste (x_q), e duas diferentes fórmulas podem ser utilizadas para converter essa distância

em peso [VIJAYARANI1; MUTHULAKSHMI, 2013], identificadas por:

$$\omega_i = \frac{1}{d(x_q, x_i)^2} \tag{3.33}$$

ou

$$\omega_i = 1 - d(x_q, x_i) \tag{3.34}$$

Desta maneira, o algoritmo do IBK é representado pelo seguindo pseudo-código:

Algorithm 1 Algoritmo IBK

1: **procedure** ALGORITMO DE TREINAMENTO
2: - Para cada exemplo de treinamento $< x, f(x) >$, adicionar o exemplo na lista "exemplos de treinamento"
3: **procedure** ALGORITMO DE CLASSIFICAÇÃO
4: - Dada uma instância requerida x_q a ser classificada,
- Sendo $x_1, ..., x_k$ denota as k instâncias da fila "exemplos de treinamento"que são mais próximas à x_q
- Retorna

$\hat{f}(x_q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
onde $\delta(a, b) = 1$ se $a = b$, caso contrário $\delta(a, b) = 0$

3.4.2.2. Rede neural artificial

Uma rede neural artificial (RNA) é essencialmente uma multidão de elementos computacionais não lineares, chamados de neurônios [MCCULLOCH; PITTS, 1943], organizados como uma rede [RUSSELL; NORVIG, 2009], e que lembra o modo como os neurônios de um ser humano são interconectados no cérebro [GONZALEZ; WOODS, 2006]. É utilizada como um desenvolver de coeficientes que age de maneira adaptativa, e representam funções de decisão através de apresentações sucessivas de conjuntos de treinamento de um determinado padrão [GONZALEZ; WOODS, 2006], isto é, funcionam como classificador em que suas capacidades de realizar previsões são empiricamente testadas [WEISS; KULIKOWSKI, 1991]. Uma RNA é composta de nós, ou unidades, conectadas por elos diretos [RUSSELL; NORVIG, 2009]. Um elo que conecta um nó j à um nó i é utilizado para propagar uma ativação a_j, isto é, uma informação que vai de j para i. Um peso $W_{j,i}$ é atribuído ao elo, que determina a força do sinal de conexão entre os

nós. Cada nó i calcula a soma de pesos das entradas,

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \tag{3.35}$$

e em seguida uma função de ativação g é aplicada à soma para se obter a saída,

$$g(in_i) = g(\sum_{j=0}^{n} W_{j,i} a_j) \tag{3.36}$$

A função de ativação utilizada neste projeto é a função não linear sigmoide, $1/(1+e^{-x})$, que tem como função ativar ou desativar um nó quando uma entrada é fornecida, isto é, atribuir o valor 1 para uma entrada correta e 0 para uma entrada errada. A representação final de um neurônio é descrita pela figura 3.6.

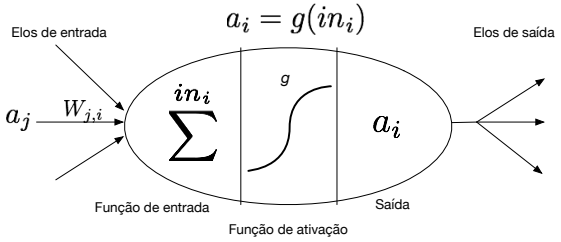


Figura 3.6: Um simples modelo matemático para um neurônio, adaptado de [RUSSELL; NORVIG, 2007]

A mais simples rede neural possível é chamada de perceptron de saída simples (também conhecido por *adaline* ou unidade de limiar lógica), e é utilizada para decidir quando um padrão de entrada pertence a uma de duas classes [WEISS; KULIKOWSKI, 1991] [GONZALEZ; WOODS, 2006]. Como é equivalente a um discriminante linear, sua representação pode ser feita por uma função de somatórios de pesos, como a seguinte equação:

$$O = \begin{cases} 1 & \text{if} \quad \sum_i w_i I_i + \theta > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.37}$$

onde a quantidade de entradas I_i é limitada e corresponde às variáveis ou características de aplicação, θ é o limiar utilizado, os pesos w_i são constantes e O é a saída, sendo que a figura 3.7 representa este perceptron.

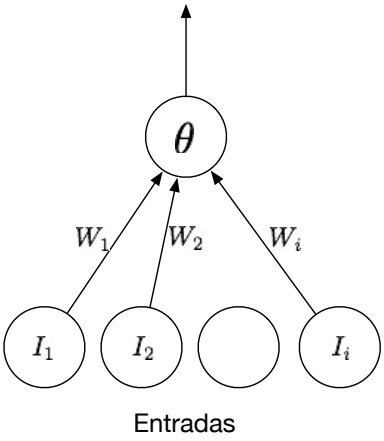


Figura 3.7: Forma geral de um perceptron de saída simples, adaptado de [WEISS; KULIKOWSKI, 1991]

A ideia básica de um algoritmo de aprendizagem de uma RNA, é ajustar os pesos da rede para minimizar o erro no conjunto de treinamento [RUSSELL; NORVIG, 2007]. Um método bastante utilizado é o sistema de aprendizado por *least mean square* (LMS), proposto por [WIDROW; HOFF, 1960], cuja saída apesar de ser representada pela equação 3.37, possui valor real. O algoritmo de LMS é configurado para minimizar o valor instantâneo da função de custo [HAYKIN, 2008], dado por:

$$E(\hat{w}) = \frac{1}{2}e^2(n) \tag{3.38}$$

onde $e(n)$ é o erro do sinal medido no tempo n. Ao diferenciar $E(\hat{w})$ em relação a $\hat{w}$,

$$\frac{\partial E(\hat{w})}{\partial \hat{w}} = e(n)\frac{\partial e(n)}{\partial w} \tag{3.39}$$

e como o LMS age em um neurônio linear, o erro pode ser expresso como:

$$e(n) = d(n) - x^t(n)\hat{w}(n) \tag{3.40}$$

onde $x(n)$ é o vetor de entrada, e como

$$\frac{\partial e(n)}{\partial \hat{w}(n)} = -x(n) \quad \textbf{e} \quad \frac{\partial E(\hat{w})}{\partial \hat{w}(n)} = -x(n)e(n) \tag{3.41}$$

e utilizando as equações 3.41 como estimativa instantânea do vetor de gradiente, podemos escrever:

$$\hat{g}(n) = -x(n)e(n) \tag{3.42}$$

O método de ajustes sucessivos aplicado no vetor de pesos w é o descendente por degrau (*steepest descent*, SD), e este é aplicado em direção contrária ao vetor de gradiente $\nabla E(w)$, resultando em:

$$g = \nabla E(w) \tag{3.43}$$

O algoritmo de SD pode ser formalmente descrito como

$$w(n+1) = w(n) - \eta g(n) \tag{3.44}$$

onde η é chamado de tamanho do degrau, ou taxa de aprendizado.

Por fim o algoritmo de LMS com o método de SD pode ser descrito como:

$$\hat{w}(n+1) = \hat{w}(n) + \eta x(n)e(n) \tag{3.45}$$

onde o inverso do parâmetro de taxa de aprendizagem η age como memória do algoritmo de LMS. O pseudo-código do algoritmo é apresentado a seguir [HAYKIN, 2008],

Algorithm 2 Pseudo-código do algoritmo de LMS

Amostras de treinamento
Vetor do sinal de entrada = $x(n)$
Resposta desejada = $d(n)$
Parâmetro selecionado pelo usuário: η
Inicialização: $\hat{w}(0) = 0$
    Computação. Para $n = 0, 1, 2...$, computar
    $e(n) = d(n) - x^t(n)\hat{w}(n)$
    $\hat{w}(n+1) = \hat{w}(n) + \eta x(n)e(n)$

3.4.2.3. Perceptron de multicamadas

Tanto o algoritmo de LMS quanto uma RNA de saída simples são sistemas de aprendizagem que descrevem apenas uma camada de RNA, consequentemente são sistemas de aprendizagem linear. Uma RNA multicamadas é formada de vários perceptrons interconectados, e se destacam por [HAYKIN, 2008]:

- o modelo de cada neurônio na RNA inclui uma função de ativação não linear que é diferenciável;
- a rede possui uma ou mais camadas que são escondidas das camadas de entrada e saída;
- a rede exibe um alto grau de conectividade, cuja extensão é determinada pelos pesos sinápticos da rede.

Uma RNA de multi-camadas pode ser estendida a um número infinito de camadas adicionais, e sua forma geral é apresentada pela figura 3.8.

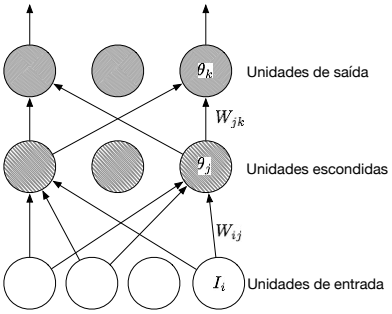


Figura 3.8: Forma geral de uma rede multicamada, adaptado de [WEISS; KULIKOWSKI, 1991]

Um método popular de treinamento de uma RNA de multicamadas é o algoritmo de propagação retroativa (*backpropagation*), que é uma generalização do algoritmo de LMS.

3.4.2.4. Algoritmo de propagação retroativa

O algoritmo de propagação retroativa, cujo termo foi popularizado após a publicação de [RUMELHART et al., 1986], foi um marco na teoria de rede neural, pois proveu uma melhor eficiência computacional aos sistemas de treinamento [HAYKIN, 2008]. O treinamento é divido em duas fases, uma direcionada para frente e uma para trás:

- quando direcionado para frente, os pesos sinápticos são fixos e o valor de entrada é propagado até a saída, passando por todas as camadas;
- quando direcionado para trás, o sinal de erro é produzido comparando o sinal de saída com o resultado esperado. Em seguida o erro é propagado através da rede, mas em sentido contrário ao inicial, e por fim o peso sináptico é ajustado.

A figura 3.9 representa a direção dos sinais.

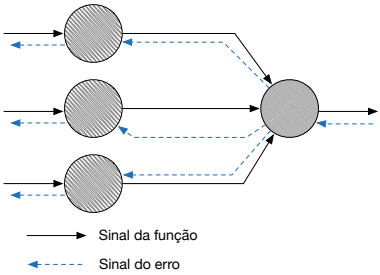


Figura 3.9: Ilustração das direções de dois sinais básicos em um perceptron de multicamadas: propagação para frente do sinal de função e para trás do sinal de erro, adaptado de [HAYKIN, 2008]

O método de propagação retroativa, como demonstrado por [RUSSELL; NORVIG, 2007], se inicia nos nós de saída, cuja regra de adaptação dos peso para uma única saída é dada por:

$$E = \frac{1}{2}\sum_{k}(y_k - \theta_k)^2, \tag{3.46}$$

onde y corresponde ao vetor de saída. Para se obter o gradiente relacionado ao peso W_{jk}, a expansão da função de ativação θ_k é realizada,

$$\begin{aligned}\frac{\partial E}{\partial W_{jk}} &= -(y_k - \theta_k)\frac{\partial \theta_k}{\partial W_{jk}} = -(y_k - \theta_k)\frac{\partial g(in_k)}{\partial W_{jk}} \\ &= -(y_k - \theta_k)g'(in_k)\frac{\partial in_k}{\partial W_{jk}} = -(y_k - \theta_k)g'(in_k)\frac{\partial}{\partial W_{jk}}(\sum_{j}^{0} W_{jk}\theta_k) \\ &= -(y_k - \theta_k)g'(in_k)\theta_j = -\theta_j \Delta_k\end{aligned} \tag{3.47}$$

sendo que $\Delta_k = Err_k x g'(in_k)$. Para se obter o gradiente relacionado ao peso W_{ij} é necessário manter a soma realizada em k, pois o processo passa retroativamente por todas as camadas. Desta forma o cálculo é

feito expandindo θ_j:

$$\begin{aligned}
\frac{\partial E}{\partial W_{ij}} &= -\sum_k (y_k - \theta_k)\frac{\partial \theta_k}{\partial W_{ij}} = -\sum_k (y_k - \theta_k)\frac{\partial g(in_k)}{\partial W_{ij}} \\
&= -\sum_k (y_k - \theta_k) g'(in_k)\frac{\partial g(in_k)}{\partial W_{ij}} = \sum_k \Delta_k \frac{\partial}{\partial W_{ij}}(\sum_j W_{jk}\theta_j) \\
&= -\sum_k \Delta_k W_{jk}\frac{\partial \theta_j}{\partial W_{ij}} = -\sum_k \Delta_k W_{jk}\frac{\partial g(in_j)}{\partial W_{ij}} \\
&= -\sum_k \Delta_k W_{jk} g'(in_j)\frac{\partial in_j}{\partial W_{ij}} \\
&= -\sum_k \Delta_k W_{jk} g'(in_j)\frac{\partial}{\partial W_{ij}}(\sum_i W_{ij} I_i) \\
&= -\sum_k \Delta_k W_{jk} g'(in_j) I_i = -I_i \Delta_j
\end{aligned} \tag{3.48}$$

O processo de propagação retroativa pode ser resumido por [RUSSELL; NORVIG, 2009]:

Algorithm 3 Algoritmo de aprendizagem da propagação retroativa

Retorna uma rede neural
Entradas: *exemplos*, um conjunto de exemplos com entrada *vetor* x e saída *vetor* y, uma rede multicamadas com L camadas, pesos $W_j i$ e função de ativação g.
Repetir
  Para cada e em *exemplos* **fazer**
    Para cada nó j na camada de entrada **fazer** $a_j < -x_j[e]$
    Para $l = 2$ até L **fazer**
      $in_k < -\sum_j W_{jk}\theta_j$
      $\theta_k < -g(in_k)$
  Para cada nó k na camada de saída **fazer**
    $\Delta_k < -g'(in_k)(y_k[e] - \theta_k)$
  Para cada $l = L - 1$ até 1 **fazer**
    Para cada nó j na camada l **fazer**
      $\Delta_j < -g(in_j)\sum_k W_{jk}\Delta_k$
      Para cada nó k na camada $l + 1$ **fazer**
        $W_{jk} < -W_{jk} + \alpha \times \theta_j \times \Delta_k$
Até que algum critério de parada seja satisfeito
Retorna uma hipótese de rede neural

3.4.2.5. Validação cruzada para k pastas

Um problema fundamental da estatística, aprendizado de máquina, redes neurais, e áreas relacionadas é que obter uma estimativa acurada para a capacidade de generalização de um algoritmo de aprendizagem em um conjunto de dados finitos [KEARNS; RON, 1997]. A validação cruzada para k pastas (*k-cross validation*) tem como objetivo estimar um valor próximo ao valor verdadeiro da generalização do erro. A utilização deste método é comum para estimar a qualidade de uma rede neural.

O processo, como apresentado por [KOHAVI, 1995], se inicia dividindo aleatoriamente um conjunto conhecido $D_(i)$ em k partes de aproximadamente mesmo tamanho, resultando em subconjuntos (pastas) $D_1, D_2, ..., D_k$. Para cada vez $t \in \{1, 2, ..., k\}$, é feito o treino em D/D_t e testado em D_t.

Sendo $x_i = (x_i, y_i)$ as instâncias do conjunto $D_(i)$, a acurácia do sistema é medida pela razão entre o número correto de classificação e o número total de instâncias do conjunto, e sua estimativa é dada por:

$$acc = \frac{1}{n} \sum_{v_i, y_i \in D} \delta(\mathcal{L}(D/D_(i), v_i), y_i) \tag{3.49}$$

onde n é o valor indexado dos subconjuntos. Ao se repetir por várias vezes utilizando diferentes subconjuntos, isto é, embaralhando as instâncias a cada novo processo, uma melhor estimativa de Monte Carlo é provida.

3.5. SISTEMA MULTIAGENTE

Uma arquitetura de referência para a implementação de sistemas multiagente (SMA) na configuração e monitoramento da PPS foi proposta por [ROLOFF, 2009], utilizando como modelo de referência o *framework* JaCaMo, desenvolvido por [BOISSIER et al., 2013], que é uma combinação de: Jason, para programar agentes; Cartago, para programar artefatos de ambiente; e Moise, para programar a organização multi-agente.

Esta arquitetura foi aplicada no Laboratório-Fábrica LabElectron, onde 4 tipos de PCIs são produzidas por uma PPS. A figura 3.10 apresenta o contexto multidimensional deste sistema.

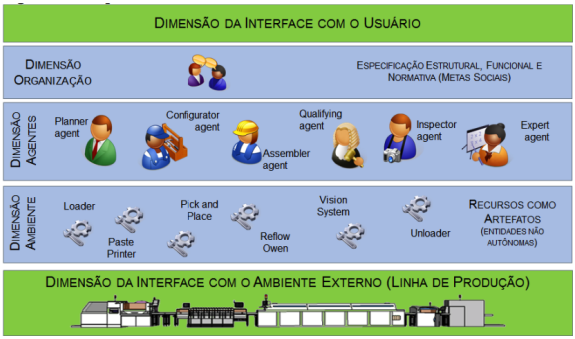


Figura 3.10: Contexto multidimensional, de [ROLOFF, 2009]

As máquinas da linha de produção são controladas e monitoradas por um sistema SCADA, e a integração entre o SMA e a linha de produção é feita através de um *Web Service*, cujo servidor é implementado utilizando o ScadaBR (software livre para sistema supervisório). Desta forma, as máquinas se comportam como artefatos (padrão de recurso) do SMA, isto é, unidades não autônomas, com o objetivo de fornecer informações referente à linha de produção para os agentes.

Dentre os agente que controlam o sistema, existe o agente *inspector*, cujo objetivo é receber diagnósticos da máquina de inspeção visual automática (que é um artefato). Este diagnóstico deve apresentar o status da inspeção, indicando os defeitos existentes. A partir do resultado obtido, o agente *inspector* se comunica com o agente de estatística para avaliar os defeitos encontrados, e quando necessário realiza a comunicação com o agente especialista, a fim de alterar a configuração dos equipamentos.

Um caminho natural de incorporar um sistema pré-existente em um moderno sistema distribuído de informação é agentificá-lo [RIBEIRO, 2003], e este processo consiste em adicionar uma camada ao sistema de forma que possibilite a comunicação entre agentes [WOOLDRIDGE, 1997], desta forma a funcionalidade do software pré-existente pode ser estendida para trabalhar com outros componentes de outros softwares [WOOLDRIDGE; JENNINGS, 1998]. Segundo [GENESERETH; KETCHPEL, 1994], existem três diferentes abordagens para a agentificação de sistemas pré-existentes:

- tradutor (*transducer*): atua como um mediador entre um programa existente e outros agentes;
- invólucro (*wrapper*): implementação de uma camada adicional para o programa existente;

- reescrever (*rewrite*): reescrever o programa adicional.

Este projeto utiliza o método de tradução, que funciona como um mediador entre o agente, que utiliza uma linguagem de comunicação entre agentes (*Agents Communication Language*, ACL), e o software nativo [RIBEIRO, 2003]. A comunicação acontece via troca de mensagens, logo é necessário que o tradutor conheça todas as frases de ambos ambientes. A agentificação traz consigo as seguintes propriedades [WOOLDRIDGE, 1997] [BORDINI et al., 2007]:

- autonomia: pode tomar decisões do que fazer baseado em um estado encapsulado, sem qualquer intervenção de humanos ou outros;
- reatividade: agentes que estão situados em um ambiente são aptos a perceber este ambiente, além de poder criar respostas de acordo com estas mudanças;
- pró-atividade: além de responderem ao ambiente, agentes são capazes de exibir um comportamento direcionado a um objetivo através de tomada de iniciativa;
- habilidade social: agentes interagem com outros agentes através de alguma linguagem de comunicação inter-agentes.

Capítulo 4

SISTEMA DE INSPEÇÃO VISUAL DE COMPONENTES DO TIPO SMD E PROTOCOLO DE COMUNICAÇÃO MÁQUINA-AGENTE

O presente trabalho visa o desenvolvimento de um sistema de inspeção óptico de componentes do tipo SMD no contexto de produção em pequenas séries, assim como a integração do sistema de inspeção ao sistema de manufatura de coordenação multiagente, viabilizado por um protocolo de comunicação. Os componentes analisados não possuem qualquer tipo de marcação ou serigrafia, e a inspeção é realizada após a montagem destes componentes na PCI, contudo antes do processo de soldagem. Como apresentado pela figura 4.1, a máquina de inspeção (S2iAOI) é controlada por um computador, cuja interface com o usuário é realizada através do software controlador. O sistema de inspeção inserido no software controlador é composto por um algoritmo de inspeção (que realiza a inspeção de fato) e por um protocolo de comunicação UDP, que fornece informações referente ao processo de inspeção ao artefato de inspeção, que por sua vez disponibiliza estas informações aos agentes. O algoritmo de inspeção proposto possui uma abordagem clássica, composto por uma etapa de pré-processamento, seguido da extração de características e avaliação do componente. As técnicas de pré-processamento têm como objetivo possibilitar a extração de características relacionadas ao formato e posicionamento do componente, através do algoritmo de região de interesse (ROI). Na etapa de extração de características, índices derivados do histograma e descritores de fourier são calculados a partir da imagem resultante da etapa de pré-processamento, formando dois vetores que em conjunto representam cada tipo de componente de maneira única. O sistema de avaliação verifica a

existência do componente em teste utilizando um classificador KNN, e o posicionamento, orientação e tipo do componente são classificados por uma RNA, por meio da técnica de *backpropagation* de multicamadas.

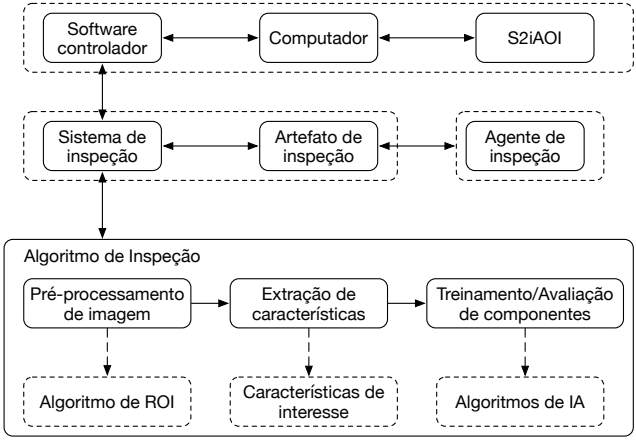


Figura 4.1: Representação geral da solução de inspeção visual de componentes do tipo SMD.

Os tópicos seguintes abordam a análise de requisitos do sistema de inspeção proposto, assim como do protocolo de comunicação de integração entre os sistemas de inspeção e coordenação multiagente.

4.1. ANÁLISE DE REQUISITOS

Os requisitos do sistema foram levantados a partir da necessidades encontradas na linha de produção do LABelectron, e melhorias propostas no projeto BRAGECRIM 013/09. Os requisitos funcionais e não-funcionais são descritos a seguir, assim como a forma de atuação no sistema desenvolvido.

4.1.1. Requisitos funcionais

RF-1. Escopo do sistema de inspeção

A meta do sistema é inspecionar os componentes posicionados na PCI durante produção de forma individual, ou seja, apesar das placas serem produzidas em painel ou individualmente, cada componente é analisado de maneira independente, isto é, para cada componente é

retirada uma imagem, desta forma é indiferente o estilo de produção adotado para o método de inspeção proposto. O software de inspeção deve ser desenvolvido em forma de plugin, desta forma pode ser integrado ao sistema de coordenação da máquina de inspeção já existente.

RF-2. Tipos de defeito

Os tipos de defeitos que o sistema de inspeção deve identificar são: componente ausente, componente descolado, componente errado e componente angulado.

RF-3. Comunicação máquina-agente

Deve existir a integração entre o software controlador S2iAOI, assim como os plugins de inspeção, com o sistema multiagente, para que a inspeção automática de SMD seja coordenada pelo MAS. Como o desenvolvimento do software de coordenação da máquina e dos plugins de inspeção foram feitos em linguagens orientada ao objeto (Java e C++), isto é, não nativa para a programação de agentes, o protocolo de comunicação a ser desenvolvido deve prover a troca de mensagens entre o agente, que representa o sistema de inspeção, e o controlador S2iAOI.

RF-4. Compatibilidade com o sistema controlador

O sistema controlador da S2iAOI realiza as funções necessárias para o funcionamento do hardware implementado na máquina de inspeção, com isso, tanto o software de inspeção quanto o protocolo de comunicação entre máquina e sistema de coordenação multiagente devem ser integráveis ao sistema controlador. Desta forma, a linguagem de programação utilizada deve ser compatível à do software controlador, assim como os acessos necessários de projetos e imagens disponíveis ao sistema multiagente e ao plugin de inspeção.

4.1.2. Requisitos não-funcionais

NF-1. Tempo de setup

O tempo de setup de inspeção de PCI para produção em larga escala toma geralmente de 3h a 1,5 dia [SZYMANSKI, 2014], desta forma o tempo para inspeção na PPS deve ser consideravelmente menor. O setup consiste em criar um novo projeto no software de coordenação, calibrar a máquina através das marcas fiduciais, selecionar os componentes de referência para treinamento e extração de características e realizar treinamento da RNA e do KNN. O tempo do treinamento varia de acordo com a quantidade de componentes cadastrados e parâmetros utilizados na RNA, contudo o tempo tomado não é longo, pois é utilizado um vetor de características da imagem, e não todos os pixels relativo ao componente.

NF-2. Tempo de inspeção

O tempo de inspeção durante PPS depende da quantidade de componentes a serem inspecionados na PCI, e varia de acordo com a velocidade de movimentação do eixo da máquina, tempo de aquisição de imagem e método de inspeção utilizado. O tipo de verificação dos defeitos é feito pela classificação do componente dentre os grupos criados, tanto da RNA quanto do KNN, e a inspeção é feita de maneira sequencial. Desta forma, para a maioria dos defeitos encontrados, a inspeção não se realiza por completo.

NF-3. Confiabilidade de inspeção

A confiabilidade de inspeção em uma PPS deve existir desde a primeira placa a ser avaliada, deste modo deve haver uma certa redundância por parte dos métodos de pré-processamento de imagem, para que não haja falha na extração de características.

4.2. RECURSOS UTILIZADOS

O software de coordenação da máquina S2iAOI foi feito em Java, com isso tanto o sistema de inspeção quanto o protocolo de comunicação agente-S2iAOI devem ser compatíveis com o software de coordenação existente. Desta forma, o plugin referente ao sistema de inspeção foi desenvolvido em Java, enquanto o protocolo responsável pela agentificação utiliza as linguagens Java e Jason (um interpretador baseado em Java para uma versão estendida do *AgentSpeak*, por [BORDINI et al., 2007]). Os processos pré-processamento de imagens e extração de características foram desenvolvidos baseado na versão 2.4.8 (lançado em dezembro de 2013) da biblioteca de processamento de imagens OpenCV. O motivo pelo qual foi escolhido o OpenCV, é a utilização da biblioteca no software de coordenação da máquina e a grande quantidade existente de algoritmos e interfaces de comunicação com dispositivos periféricos. Para o processo de classificação baseada no algoritmo de KNN, foi utilizada a biblioteca Weka (*Waikato Environment for Knowledge Analysis*, desenvolvido pela Universidade de Waikato, [HALL et al., 2009]), também em Java. Além do uso da biblioteca do Weka para o desenvolvimento do sistema de inspeção, o software do Weka foi utilizado para realizar as validações cruzada e relatórios referentes ao KNN e à RNA. A biblioteca Neuroph, um framework de rede neural desenvolvido em java, foi utilizada para desenvolver o MLP e o sistema de backpropagation. Apesar do OpenCV oferecer pacotes referentes ao KNN e à RNA, as bibliotecas utilizadas se mostram mais eficientes, pois apresentaram menor custo computacional, justificando as escolhas.

Imagens para fins didáticos foram feitas utilizando o Matlab 2014b, e toda a programação foi feita em um MacBook Pro com o sistema operacional OS X Yosemite, logo todo o código desenvolvido é compatível ao software de coordenação da máquina, este rodando em Linux Ubuntu 12.04.

4.3. ESTRUTURA DO SOFTWARE DE INSPEÇÃO

O software de inspeção desenvolvido neste trabalho foi criado de forma que haja uma integração completa com o software implementado na máquina S2iAOI, desta foram utilizadas classes em java para realizar a inspeção, e como o software da S2iAOI já possui um *main*, as classes criadas devem apenas ser inclusas e chamadas pelo *main*. A figura 4.2 representa o diagrama de classes utilizando elementos UML (*Unified Modeling Language*).

As funcionalidades de cada classe são:

- Carac: concentra todas as informações referente aos componentes *gold*, incluindo endereços no diretório, informações e características;
- Crop: recorta uma imagem, de componente *gold*, para delimitar a área de busca no algoritmo de ROI;
- Autocrop: recorta uma imagem de componente *gold* utilizando o quadro obtido pela ROI;
- Features: A partir do quadro recortado pela classe Crop, são identificadas as características relacionadas ao formato e posição do componente *gold*;
- Histogram: calcula o histograma das imagens *gold*;
- NNO2: treina a MLP utilizando o método de backpropagation;
- gold: controla a criação de arquivos e instâncias derivadas de imagens *gold*;
- Caractest: concentra todas as informações referente aos componentes em teste, incluindo endereços no diretório, informações e características;
- Rotationtest: verifica se há rotação do componente teste quando comparado ao componente *gold* respectivo;
- testImages: controla a criação de arquivos e instâncias derivadas de imagens de componentes em teste;
- Croptest: recorta uma imagem de componente teste, utilizando o mesmo quadro da imagem *gold* respectiva;
- Autocroptest: recorta uma imagem de componente teste, utili-

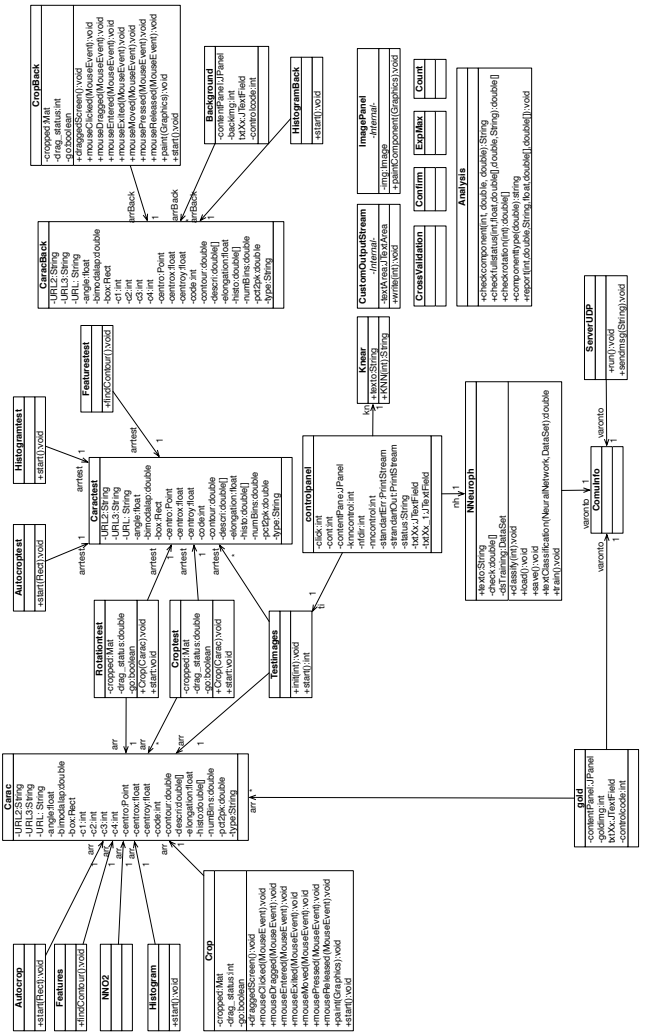


Figura 4.2: Diagrama de classes do software de inspeção.

zando a mesma ROI da imagem *gold* respectiva;

- Histogramtest: calcula o histograma das imagens de componentes em teste;
- Featurestest: a partir do quadro recortado pela classe Crop, são identificadas as características relacionadas ao formato e posição

do componente em teste;
- ControlPanel: cria um painel de controle utilizado para realizar a inspeção;
- NNeuroph: classifica os componentes em teste, salva e carrega as configurações da rede neural e testa classificação utilizando a técnica de validação cruzada para K-pastas;
- Comuinfo: monitora o status dos processos do software de inspeção;
- ServerUDP: cria um servidor UDP (*User Datagram Protocol*) para possibilitar a comunicação entre o software de inspeção e o agente que representa a máquina S2iAOI;
- Knear: cria o sistema de classificação KNN, e classifica os componentes em teste;
- CaracBack: concentra todas as informações referente as imagens de fundo de placa, incluindo endereços no diretório, informações e características;
- CropBack: recorta uma imagem do fundo de placa, para delimitar a área utilizada para se obter o histograma;
- Background: controla a criação de arquivos e instâncias derivadas de imagens de fundo de placa;
- HistogramBack: calcula o histograma das imagens de fundo de placa;
- Analysis: concentra o resultado de todos os processos que avaliam um componente em teste, gerando uma análise final;
- CrossValidation: realiza a validação cruzada de k-pastas para o algoritmo de KNN;
- ImagePanel: cria os painéis de controle para o funcionamento do software de inspeção;
- Confirm: cria janela de confirmação para diversos processos, servindo como bloqueio para outros processos não ocorram paralelamente de modo indevido;
- Count: monitora a quantidade de imagens criadas pelo software de gerenciamento da máquina em um projeto;
- CustomOutputStream: sistema para informar ao usuário quais processos estão ocorrendo, substituindo o *System.out.print* padrão do Java.

Existe a interação entre algumas classes, mas outras utilizam informações salvas em arquivos, possibilitando parar e recomeçar a inspeção após todas as imagens *gold* serem devidamente cadastradas. A figura 4.3 mostra o diagrama de interação entre classes.

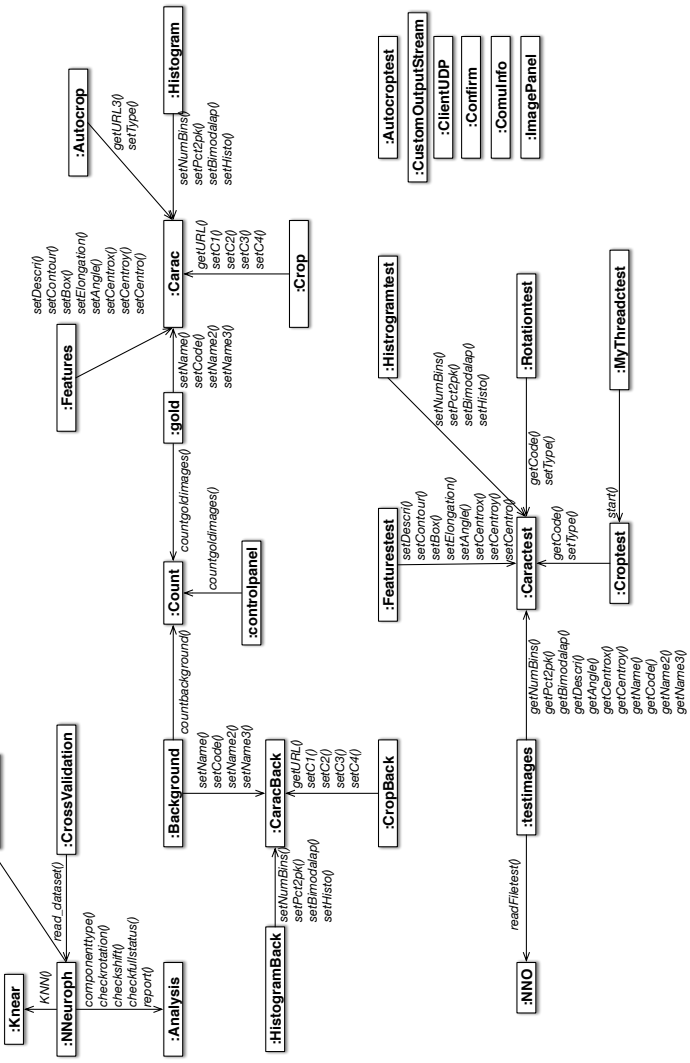


Figura 4.3: Diagrama de interação entre classes.

Os arquivos utilizados e criados pelo software de inspeção seguem o padrão proposto por [MELO, 2013], cujo projeto criado é denominado PCB_ID**V*, e todas as imagens obtidas pela máquina S2iAOI possui dimensão 640x480 pixels. As pastas e arquivos de cada projeto são

organizados como mostra a figura 4.4.

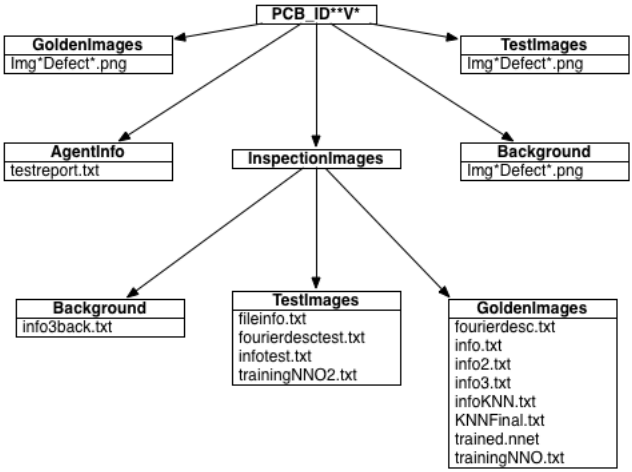


Figura 4.4: Organização de pastas e arquivos.

As imagens referente a componentes *gold* são salvas dentro da pasta *Gold* e nomeadas seguindo o padrão $Img*Defect*.png$, enquanto as imagens de componentes em teste são salvas na pasta $TestImages$, mantendo o mesmo padrão de denominação utilizado pelas imagens *gold*. Para as imagens de fundo de placa, é criada uma pasta chamada *Background*, também utilizando o mesmo padrão de nomeação para as imagens. As informações que devem ser disponibilizadas ao agente, são salvas no arquivo *testreport.txt*, salvo este na pasta AgentInfo. Uma pasta chamada $InspectionImages$ é criada para salvar os arquivos gerados (em formato txt) ao longo de todo o processo de inspeção, sendo subdivido em pastas referentes a componente *gold*, em teste/testados e fundo de tela. Para as imagens *gold*, os arquivos gerados são:

- fourierdesc: contém os descritores de Fourier;
- info: informações da primeira extração de características, referentes ao tipo de componente, contorno e dados para próximas etapas;
- info 2: informações da segunda extração de características, referentes à posição e ângulo do componente;
- info 3: informações da terceira extração de características, referentes ao histograma;
- infoKNN: padronização de informações para agilizar o processa-

mento do KNN;
- trainingNNO: padronização de informações para o treinamento da MLP;
- trained.nnet: arquitetura salva da MLP utilizada na inspeção do projeto corrente.

Para as imagens teste, são criados os seguintes arquivos, que são utilizados como entrada em alguns processos e arquivo de log ao fim da inspeção completa:
- fourierdesctest: contém os descritores de Fourier;
- fileinfo: informações da primeira extração de características, referentes ao tipo de componente, contorno, posição e ângulo;
- infotest : informações da segunda extração de características, referentes ao histograma;
- trainingNNO2: padronização de informações para a classificação do componente pela MLP;

As imagens de fundo de placa utilizam somente as informações derivadas do histograma, desta forma, apenas o arquivo $info3back$ é criado.

4.4. FLUXOGRAMA DO FUNCIONAMENTO DO SOFTWARE DE INSPEÇÃO

Foi desenvolvido para o software de inspeção SMD deste trabalho uma arquitetura de processamento de imagens, a qual para fins didáticos divide-se em três etapas: a primeira é destinada a criar o sistema de avaliação, isto é, refere-se ao tratamento das imagens *gold* (ou de referência), a segunda utiliza as imagens referente ao fundo da PCI para servir como referência sobre a existência de componentes, e a terceira realiza a inspeção das imagens dos componentes em teste. A figura 4.5 apresenta o fluxograma geral de funcionamento do software de inspeção.

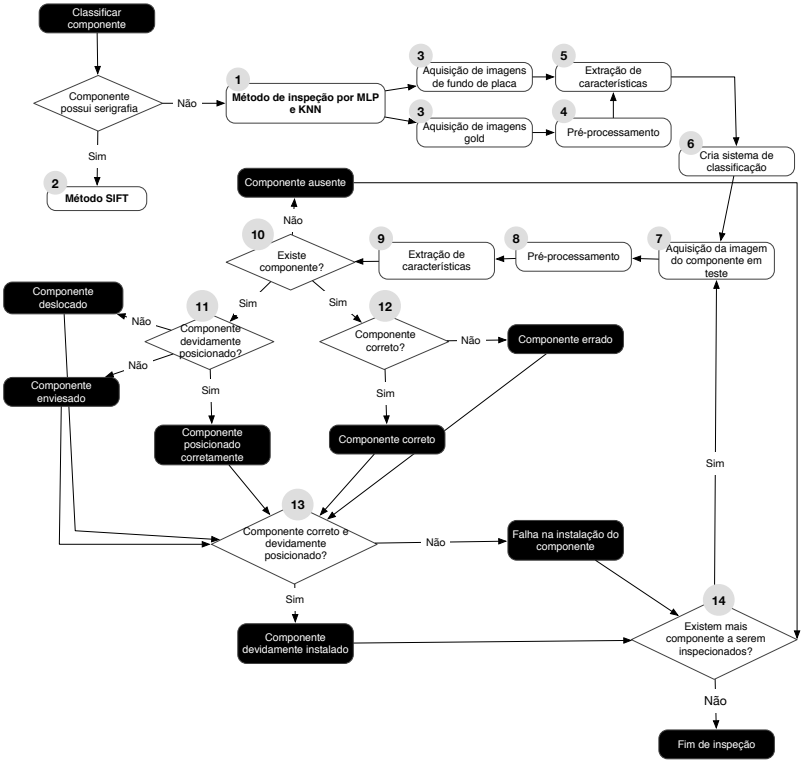


Figura 4.5: Fluxograma geral do sistema de inspeção.

A primeira etapa consiste em dividir a inspeção de componentes de acordo com a existência, ou não, de serigrafia e marcações. O método proposto por [SZYMANSKI, 2014] (2) inspeciona componentes com serigrafia e é tratado em seu trabalho de dissertação, enquanto (1) realiza a inspeção de componentes sem serigrafia e será detalhado neste trabalho.

As imagens dos componentes *gold* e imagens amostradas do fundo de tela são as referências utilizadas para a criação do sistema de classificação, e estas são adquiridas na etapa (3) no formato de quadro (*frame*), coordenada pelo sistema de gerenciamento da máquina S2iAOI e realizada pelo operador. O pré-processamento (4) é aplicado apenas em imagens de componentes, e tem como função determinar a ROI assim como fornecer um formato de dados que facilite a próxima

etapa. Duas classes de características são utilizadas como entrada nos sistemas de classificação, e cabe à etapa de extração de características (5) prover tais informações. Uma das classes é proveniente de índices derivados do histograma, e é performado nas imagens *gold* e fundo de tela partir do quadro cru, ou seja, sem pré-processamento. A outra classe é corresponde a *features* referentes a posição e formato obtidos do resultado da etapa de pré-processamento, logo realizado apenas em quadros que representam componentes. O sistema de classificação (6) é criado a partir do momento em que todas os componentes *gold* e exemplos de fundo de tela são devidamente cadastrados e suas características extraídas, constituindo dois métodos de classificação, um para detecção da existência de componente (KNN) e um para verificação do tipo de componente (MLP). Ao fim da etapa (6) o sistema está pronto para realizar a inspeção de PCIs durante produção, e a ordem de componentes a serem inspecionados segue a ordem de cadastro das imagens *gold*. Para cada nova PCI a ser inspecionada, é necessário recalibrar a máquina (feita de maneira automática guiada pelas marcas fiduciais) para que não haja variação na posição relativa dos componentes. A aquisição do primeiro componente a ser inspecionado (7) acontece ao fim do processo de recalibração, e o quadro utilizado mantém a posição e tamanho do frame da imagem *gold* correspondente. Em seguida o quadro é pré-processado (8) e as características da imagem teste são extraídas (9). A primeira verificação consiste em averiguar a existência do componente (10) através do classificador de KNN, ou seja, comprovar que existe de fato um componente instalado. Em caso da ausência, o sistema salva em um arquivo a informação de "componente ausente"e a inspeção segue para o próximo componente da lista. Na etapa (11) é comparada a posição e ângulo do componente teste com os valores do componente *gold* refente. O classificador MLP determina se o componente instalado é do tipo correto (12), e a etapa (13) avalia todas as respostas obtidas, para determinar se a instalação foi correta ou houveram falhas. Ao fim do processo, o número de componentes inspecionados (14) deve corresponder ao número de componentes *gold* cadastrados, e para cada PCI inspecionada um arquivo de log de dados é criado, salvando todas as informações obtidas ao longo do processo de inspeção de cada componente.

Nas sessões seguintes serão apresentados os recursos utilizados, a etapa de pré-processamento, extração de características, treinamento, avaliação do componente e o protocolo de integração S2iAOI-agente.

4.5. PRÉ-PROCESSAMENTO DE IMAGEM

A arquitetura de pré-processamento consiste em uma pipeline gráfica[1] para determinar a região de interesse do quadro (ROI, *Region of Interest*), isto é, definir a região que melhor representa o componente, bem como fornecer uma representação desta região por meio de um vetor 2-D, como é mostrada na figura 4.6.

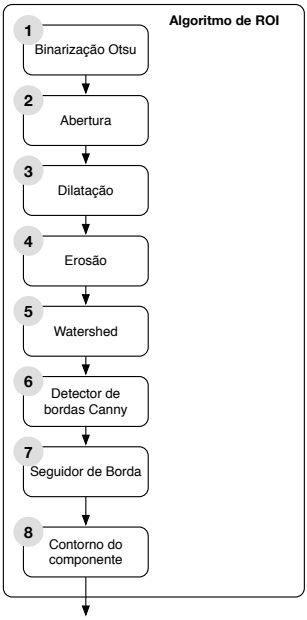
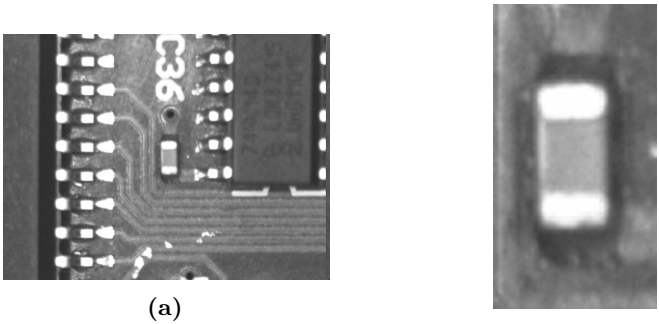


Figura 4.6: Arquitetura do sistema de pré-processamento.

Um capacitor SMD, representado pela figura 4.7a, será utilizado para exemplificar a arquitetura do pré-processamento de imagem, e como descrito anteriormente, todos os processos relativos a inspeção utilizam o quadro recortado pelo operador, e não a imagem fornecida pela câmera. Desta forma, o quadro que representa o capacitor SMD da figura 4.7a é descrito pela imagem 4.7b.

A primeira etapa do algoritmo de ROI é tornar o frame binário, pois diminui o custo computacional dos processos posteriores. Como

[1] Uma sequência de etapas utilizadas para criar uma representação 2-D de uma cena 3-D



(a) **(b)**

Figura 4.7: (a) Capacitor SMD instalado na PCI e (b) quadro do capacitor SMD.

o histograma do quadro possui característica bimodal e o objetivo é separar o fundo de tela do objeto, o método Otsu (1) é adequado para esta aplicação, contudo o resultado gera ruídos que atrapalham a identificação do componente, como é apresentado pela figura 4.8.

Figura 4.8: Resultado da imagem binarizada pelo método Otsu.

A operação morfológica de abertura (2) é aplicada para eliminar os ruídos existentes na região correspondente à superfície do capacitor. Utilizando uma estrutura de elementos no formado elipsoidal de tamanho $3x3$, como é ilustrado em 4.9a, obtém-se um componente com uma superfície sem falhas, apresentado pela figura 4.9b.

Ao aplicar a operação de dilatação (3) por três vezes (mostrou-se suficiente para esta aplicação), exclui-se as regiões que não correspondem ao componente e pequenas interconexões que possam haver entre componentes, identificando a região de presença do capacitor, como

Figura 4.9: (a) Estrutura de elementos elipsoidal e (b) resultado da operação de abertura.

é ilustrado na figura 4.10a. Consequentemente pixels pertencente à borda do componente são perdidos, e para recuperá-los a operação de erosão (4) é aplicada, na mesma quantidade da operação anterior. O resultado é uma imagem composta pelo componente e pelos demais ruídos (figura 4.10b), contudo a região que corresponde ao componente agora é conhecida.

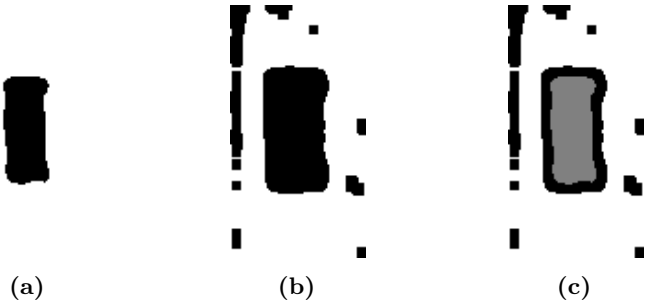


Figura 4.10: (a) Resultado da operação de dilatação, (b) resultado da operação de erosão e (c) soma das operações de dilatação e erosão.

A soma entre as operações de dilatação e erosão é uma imagem (figura 4.10c) que apresenta uma maior clareza na diferença entre o componente e o fundo de placa, cuja região em cinza corresponde ao capacitor, a região branca ao fundo da placa e as regiões em preto ainda são indefinidas. Cabe então ao algoritmo de Watershed (5) delimitar qual porção da região indefinida pertence ao componente e qual deve

ser descartada, isto é, o produto do algoritmo deve corresponder ao capacitor, que por sua vez à ROI, como é ilustrado na figura 4.11.

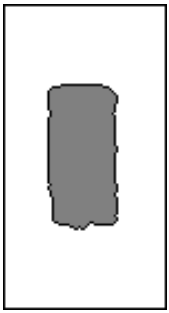


Figura 4.11: Resultado do algoritmo de Watershed.

Para alguns componentes, como o resistor SMD da figura 4.12a, o resultado do algoritmo de Watershed é inconclusivo, como representado pela figura 4.12b.

(a) **(b)**

Figura 4.12: (a) Quadro do resistor SMD e (b) resultado do algoritmo de Watershed aplicado ao resistor.

Diante desse fato, aplica-se o detector de bordas Canny (6) para verificar quantas regiões existem na imagem. Assume-se o valor 0 para o limiar inferior e 1 para o limiar superior, pois a imagem está binarizada. O resultado deste processo quando aplicado ao capacitor SMD é representado pela figura 4.13.

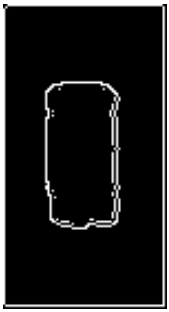
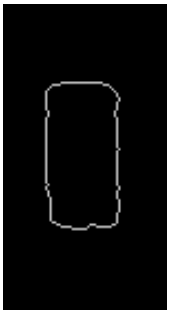
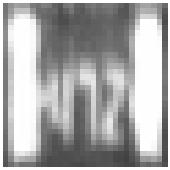


Figura 4.13: Resultado do detector de bordas Canny e do algoritmo seguidor de borda.

O algoritmo seguidor de borda (7) aplicado na imagem resultante da etapa (6) identifica os contornos que sejam fechados e mais externo o possível, resultando no perímetro de cada região identificada. Para o capacitor, o maior contorno encontrado corresponde a ROI, como mostra a figura 4.14a. Para componentes com mais de um contorno identificado, como o resistor SMD, uma análise é performada na etapa (8). Como os componentes são simétricos, uma comparação entre perímetros é realizada, e caso sejam parecidos, estes são interligados por retas. O resultado da interconexão dos perímetros por retas é comparado à forma de um quadrilátero, pois todos os componentes testados possuem este formato, e se corresponderem ao esperado, é definido como ROI. A figura 4.14b representa a ROI encontrada quando mais de dois contornos são identificados em um resistor SMD.



(a)



(b)

Figura 4.14: (a) ROI obtida a partir do quadro de um capacitor SMD. e (b) ROI obtida a partir do quadro de um resistor SMD.

4.6. EXTRAÇÃO DE CARACTERÍSTICAS DE INTERESSE

A extração de características tem como objetivo obter métricas relacionadas a uma imagem de forma que um vetor formado por essas métricas representem um objeto de maneira única. Consequentemente, um espaço menor de memória é utilizado para armazenar as informações referentes a uma imagem, aumentando a rapidez do software de inspeção. As imagens que representam o fundo da placa, são representadas por um vetor que contém apenas características derivadas do histograma, enquanto os componentes são representados por métricas derivados do histograma, contorno e posição da ROI. A figura 4.15 apresenta a árvore do processo de extração de características.

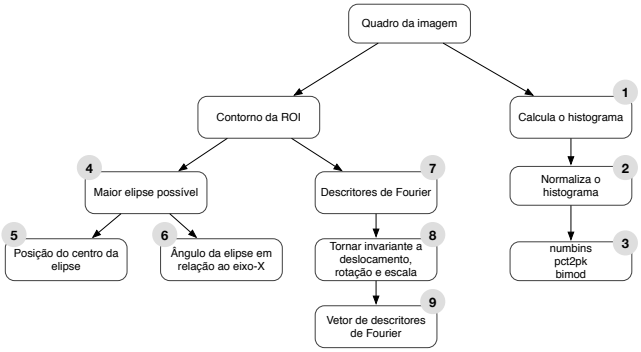


Figura 4.15: Árvore do processo de extração de características.

As características derivadas do histograma serão utilizadas para determinar a existência ou ausência de componente numa determinada posição, e as métricas utilizadas são:

- porcentagem de bins do histograma com quantidade superior a 0.5% do total de pixels (numbins);
- porcentagem da extensão do histograma entre os dois maiores picos (pct2pk), cujo pico é definido quando posicionado consecutivamente a bins entre duas mínimas e possuir uma quantidade maior que 0.5% do total de pixels;
- bimodalidade da diferença entre pixels pertencentes aos vizinhos mais próximos (bimod).

A partir do quadro obtido o histograma é calculado (1) e normalizado (2), para que as características relacionadas sejam obtidas. Para o

capacitor SMD, o histograma obtido é representado pela figura 4.16, e os valores das características (3) são extraídos no formato *double*, formando um vetor $V = [numbins, pct2pk, bimod, componente/fundodetela]$.

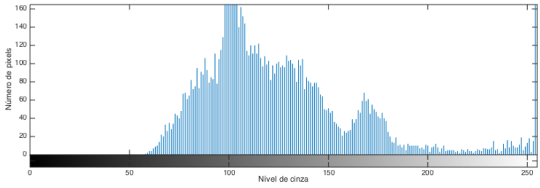


Figura 4.16: Histograma de quadro de um capacitor SMD.

As características provenientes da ROI são utilizadas para verificar se a correspondência, entre componentes *gold* e teste, referente a posição, orientação e tipo entre componentes é válida. O ângulo e ponto de referência do componente são obtidos a partir da maior elipse possível que se encaixa dentro da ROI, onde o ponto de referência (5) assume o valor da centroide da elipse, e a orientação (6) é obtida através da diferença entre o ângulo da elipse e o eixo x, como mostra a figura 4.17.

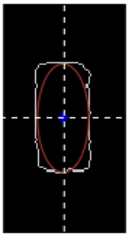


Figura 4.17: Elipse inserida na ROI de um capacitor SMD.

O conjunto de descritores de Fourier (7) formados a partir da ROI formam um vetor que representam uma aproximação da forma do contorno de um objeto, como exemplificado pela figura 4.18.

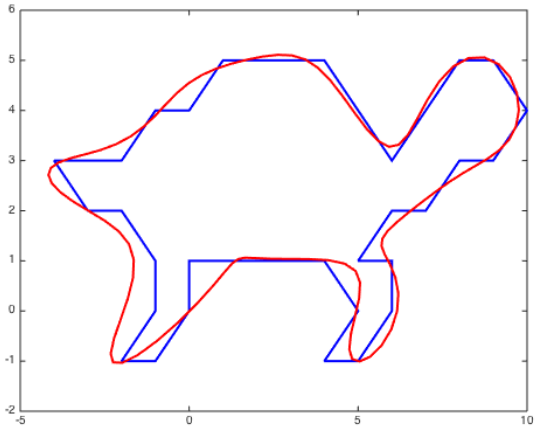


Figura 4.18: Aproximação de objeto por descritores de Fourier.

Como análise da orientação e posição é feita utilizando as características obtidas a partir da elipse, e não há variação de escala, os descritores de Fourier utilizados são invariantes a deslocamento, rotação e escala (8). Desta forma o classificador utilizado para determinar o tipo de componente se torna mais robusto. Quanto maior o número de descritores utilizados, maior a semelhança com o contorno do componente, e para este projeto um vetor formado por 6 descritores (9) mostrou-se eficaz para representar um componente de forma única.

Ao fim do processo de extração de características, para cada componente três vetores são formados para servir de entrada aos sistemas de classificação ou treinamento.

4.7. SISTEMAS DE CLASSIFICAÇÃO

Dois tipos de sistemas de classificação são utilizados neste trabalho, o KNN, que é utilizado para determinar a existência ou ausência de componente, e o backpropagation de multicamadas, que tem como objetivo verificar o tipo de componente. Para ambos sistemas de classificação não há a necessidade de utilizar exemplos negativos para o treinamento. As imagens *gold* e de fundo de placa são utilizadas como entrada para o treinamento, e estes acontecem em *threads* exclusivas.

4.7.1. K nearest neighbor

O algoritmo de KNN utiliza os vetores de características derivadas do histograma como instâncias para criar um sistema de classificação não paramétrico, com o intuito de determinar a existência ou ausência de componente na posição de análise vigente. Desta forma, o classificador possui dois *clusters*, e a tabela 4.1 mostra o padrão de valores esperados para cada um.

Tabela 4.1: Clusters do KNN

Clusters	**numbins (nb)**	**pct2pk (pk)**	**bimod (bd)**	**tipo**
Component	$39<x<65$	$829<x<12100$	$7<x<30$	*component*
Background	$12<x<65$	$4300<x<16000$	$4<x<8$	*background*

O espaço de busca do conjunto de características é representado pela figura 4.19, onde inicialmente é vazio e a cada imagem adquirida, deve atingir o estado final do espaço de busca.

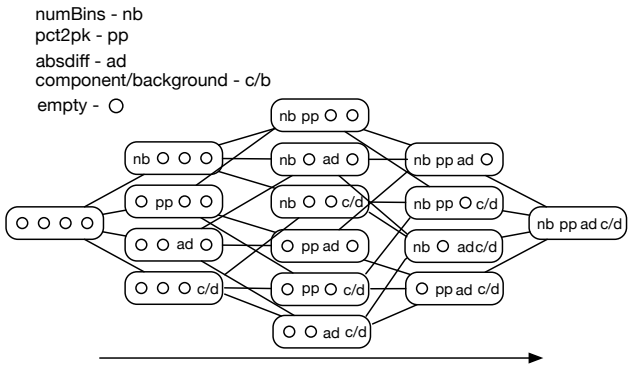


Figura 4.19: Espaço de busca do classificador KNN.

Para o treinamento do KNN são utilizadas as imagens *gold* e de fundo de tela, e o classificador é inicializado com 1 vizinho próximo. As instâncias utilizadas para o treinamento são adicionadas seguindo uma ordem FIFO (*first in, first out*), e o algoritmo de busca utilizado é linear, ou seja, é um método de força bruta.

4.7.1.1. Validação cruzada de K-pastas para KNN

A validação cruzada para KNN é utilizada, neste contexto, para definir o valor do número de pastas (k) do classificador, de forma que minimize o erro de predição. Uma função de perda é utilizada para medir este erro, onde as entradas são instâncias verdadeiras e conhecidas, e a saída retorna um valor alto quando difere da resposta esperada, e 0 para uma correspondência verdadeira. A função de perda l é definida pela equação 4.1.

$$l(tipo, h_k(nb, pk, bd| \ \{nb, pk, bd, tipo\} \quad i = 1, ..., n)) \tag{4.1}$$

onde $h_k(nb, pk, bd| \ \{nb, pk, bd, tipo\} \quad i = 1, ..., n)$ representa o método de vizinho mais próximo, e identifica os k vizinhos mais próximos das entradas nb_i, pk_i, bd_i referente ao conjunto de treinamento, retornando se é componente ou fundo de placa. Como não é possível determinar a média da perda utilizando métodos tradicionais, como a diferença entre quadrados, pois não se sabe a distribuição das instâncias, a abordagem de validação cruzada é utilizada para determinar o melhor k possível, visando minimizar a perda esperada.

O método consiste em dividir as informações conhecidas em dois grupos, um subconjunto T (que representa o treinamento) e um subconjunto V (que representa a validação). Deste modo, T são os vizinhos e V o conjunto teste, tornando-se as novas observações que necessitam ser classificadas. Com isso o erro esperado de predição pode ser aproximado por uma média, resultando na validação para o subconjunto V, representado pela fórmula 4.2.

$$\frac{1}{\#V} \sum_{x_j \in V} l(tipo_j, h_k(nb_j, pk_j, bd_j \mid \{nb, pk, bd, tipo\} \quad \in T) \tag{4.2}$$

Caracteriza-se validação cruzada quando são utilizados diferentes subconjuntos de treinamento e validação, e para caracterizar uma validação cruzada de K pastas, dividi-se o conjunto inicial de forma aleatória em K subconjuntos de mesmo tamanho. Consequentemente a validação acontece K vezes, e para a $i - ésima$ iteração, a validação cruzada é definida pela equação 4.3.

$$\frac{1}{K} \sum_{i=1}^{K} \frac{1}{\#V_i} \sum_{x_j \in V_i} l(tipo_j, h_k(nb_j, pk_j, bd_j \mid \{nb, pk, bd, tipo\} \quad \in T_i) \tag{4.3}$$

Desta forma é possível aproximar a perda esperada com a média da perda observada. Ao aplicar o método de validação cruzada para 10 pastas, utilizando 60 instâncias dividas igualmente em dois *clusters*, onde um refere-se a imagens *gold*, e a imagens do fundo de placa, resulta na validação do método para identificação da presença ou ausência de componente, com apenas 1 instância erroneamente classificada dentre as 10 pastas, como mostra a tabela 4.2.

Tabela 4.2: Acurácia do KNN

Número total de instâncias	60	
Instâncias classificadas corretamente	59	98,3333%
Instâncias classificad,as incorretamente	1	1,6667%
Estatística kappa	0,9667	
Erro absoluto médio	0,0263	
Erro médio quadrático	0,1289	
Erro absoluto relativo	5,2666%	
Erro relativo quadrático	25,7709%	

4.7.2. *Backpropagation* de multicamadas

A classificação de componentes referente ao modelo é realizada através de uma RNA com multicamadas, cujo treinamento é feito utilizando o método de *backpropagation*. O *input* para treinamento da RNA é formado pelo vetor de 6 posições contendo os descritores de Fourier e o modelo de cada componente *gold*, consequentemente a quantidade de *inputs* utilizadas para o treinamento varia de acordo com a quantidade de componentes *gold* cadastrados. Para realizar o treinamento é necessário normalizar os valores de entrada, desta forma a fórmula 4.4 é aplicada nos valores dos descritores de Fourier.

$$nv = \frac{value - (\frac{sv}{2})}{(bv + 1000) - (\frac{sv}{2})} \tag{4.4}$$

onde nv é o valor normalizado, sv é o menor valor e bv o maior valor. Como os maiores e menores valores utilizados na normalização são obtidos a partir dos descritores de imagens *gold*, e não pode haver valores maiores que 1 ou menores que 0, foi adicionado 1000 ao bv, e dividido o sv por 2, para que haja uma margem de segurança, deste modo quando utilizados descritores provenientes das imagens em teste não há risco de falha devido a valores fora da extensão prevista. A informação que diz respeito ao modelo do componente também deve

ser normalizada, e esta se encontra em forma textual, desta forma, deve atribuir aos modelos cadastrados pelas imagens *gold* valores entre 0 e 1, onde cada ponto representa um modelo de componente, como mostra a figura 4.20 para cinco diferentes modelos de componentes.

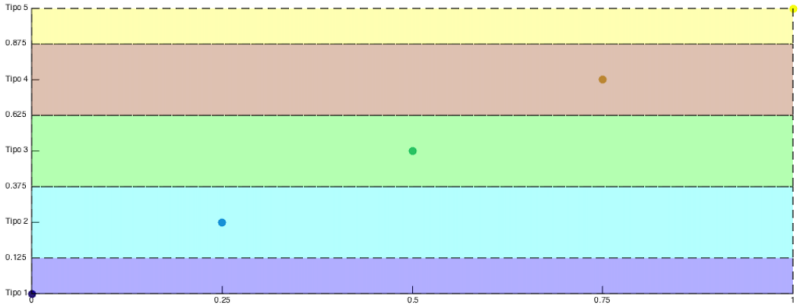


Figura 4.20: Tipos de componentes divididos pelo alcance de valor atribuído.

A RNA de multicamadas, representada pela figura 4.21, utilizada neste projeto tem as seguintes características:

- a entrada é formada por vetores normalizados de descritores de Fourier extraídos das imagens *gold*;
- possui 30 neurônios em uma camada escondida;
- a saída é o tipo do componente descrito de forma normalizada;
- função de ativação sigmoidal;
- taxa de aprendizagem de 0.4;
- momento de 0.9;
- erro máximo de 0.00001;
- 5000 épocas utilizadas para o treinamento.

4.7.2.1. Validação cruzada de K-pastas para MLP

O método de validação cruzada é utilizado como estratégia de seleção da arquitetura da MLP, e tem como objetivo evitar o *overfitting* e o *underfitting*. É possível observar a presença do *overfitting* quando o erro de validação aumenta e o erro de treinamento diminui, resultando em uma simples memorização dos dados de treinamento. O *underfitting* acontece quando os erros de validação e treinamento se mantem grandes, resultando em um aprendizado não suficiente. Como não é possível minimizar o erro esperado por algum tipo de generalização, utiliza-se novamente o processo de validação cruzada.

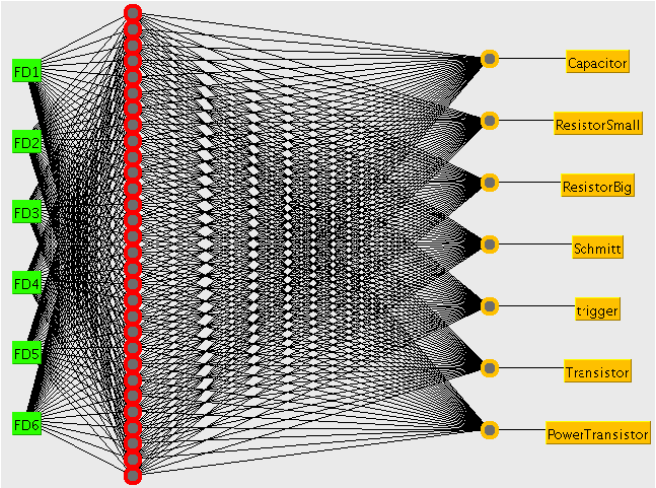


Figura 4.21: RNA de multicamadas.

Assumindo que os padrões de entrada x são retirados de uma distribuição de probabilidade P, e todos os conjuntos de informações são derivadas da função verdade f^*, os conjuntos utilizados para a validação cruzada são definidos por $\{(x^{(1)}, d^{(1)}), ..., (x^{(K)}, d^{(K)})\}$, com $x^{(i)} \sim P$ e $d^{(i)} = f^*(x^{(i)})$, resultando na aproximação do erro esperado ao erro de treinamento, como é mostrado na fórmula 4.5.

$$\frac{1}{K}\sum_{i=1}^{K}(\frac{1}{2}(f(x^{(i)}) - d^{(i)})^2) \tag{4.5}$$

Desta maneira, quando há uma boa aproximação, a hipótese aprendida através do conjunto de treinamento performará bem em todos os conjuntos de dados relacionados. Contudo a fórmula 4.5 trata apenas de um conjunto, e a função de aprendizado deve ter uma performance satisfatória independente dos dados de validação e teste. Para isso, o algoritmo de validação cruzada de K-pastas 4 é aplicado, sendo que o conjunto (D) formado pelos descritores de Fourier das imagens *gold*, é subdivido em K conjuntos disjuntos do mesmo tamanho $(D_1, ..., D_K)$, avaliando o modelo para K subgrupos.

Devido a MLP utilizar vários parâmetros de ajuste, técnicas de regularização foram utilizadas para obter um melhor desempenho do processo de validação cruzada. A técnica de decaimento de peso foi utilizada para evitar o *overfitting* e *underfitting*, e esta consiste

Algorithm 4 Pseudo-código do algoritmo de validação cruzada de K-pastas para MLP

1: **Para:** $D, 2 \leq k \leq K$
2: Resposta desejada = $d(n)$
3: **For** $i = 1$ to K
4: (Re)- inicializa a RNA
5: Treina a RNA para os subconjuntos $D_1 \cup ... \cup D_{i-1} \cup D_{i+1} \cup ... \cup D_K$
6: Calcula o erro de teste médio e_i sobre D_i
7: **End for**
8: **Retorna:** $\frac{1}{K} \sum_{i-1}^{K} e_i$

em modificar o erro quando este é minimizado longo do aprendizado, manipulando a variável λ, aplicada na fórmula 4.6.

$$E(\overrightarrow{w}; D; \lambda) := E(\overrightarrow{w}; D) + \frac{1}{2}\lambda||\overrightarrow{w}||^2 \tag{4.6}$$

Segundo a literatura, os valores de λ variam entre 0 e 10. Se o valor é considerado pequeno à aplicação em questão, gera o *overfitting*, e caso for grande, ocorre o *underfitting*.

Como foi constatado que o número de 30 camadas escondidas resulta em uma boa capacidade de representação, o mesmo valor de peso foi aplicado a todos os neurônios, resultando em um menor número de parâmetros a serem processados, que por sua vez se torna menos complexo, criando uma máscara que é ideal para casos de identificação de objetos.

Foram utilizados 30 componentes de seis modelos diferentes (5 componentes de cada modelo) para determinar a arquitetura da MLP, utilizando o método de treinamento *backpropagation*. Os componentes SMD utilizados são: capacitor, resistor tipo 1, resistor tipo 2, Schmitt Trigger, transistor de potência e transistor. Realizando o treinamento após os ajustes de parâmetros, isto é, com a configuração já apresentada pela seção backpropagation de multicamadas, o maior erro obtido foi de 0,0264577, ou seja 2,6%, como mostra a tabela 4.3.

Tabela 4.3: Erro por época

	Erro depois de 50000 épocas
Pasta 0	0,0000011
Pasta 1	0,0000012
Pasta 2	0,0000001
Pasta 3	0,0000012
Pasta 4	0,0000009
Pasta 5	0,0000012
Pasta 6	0,0000001
Pasta 7	0,0052916
Pasta 8	0,0264577
Pasta 9	0,0000001
Pasta 10	0,0052921

O $K = 10$ treinamentos realizados resultaram apenas uma instância classificada incorretamente, como mostra a tabela 4.4.

Tabela 4.4: Acurácia do treinamento por *backpropagation*

Número total de instâncias	30	
Instâncias classificadas corretamente	29	96,667%
Instâncias classificadas incorretamente	1	3,334%
Estatística Kappa	0,96	
Erro absoluto médio	0,0642	
Erro quadrático médio	0,1389	
Erro relativo absoluto	26,8225%	
Erro relativo quadrático	25,7709%	

Segundo a matriz de confusão 4.5, apenas um tipo de componente foi classificado incorretamente em todos os testes realizados, e este é um Schmitt trigger, desta forma a arquitetura proposta se mostra suficiente para avaliar os componentes SMD.

Tabela 4.5: Matriz de confusão dos componentes *gold*

a	**b**	**c**	**d**	**e**	**f**		Classificado como
5	0	0	0	0	0	**a**	Capacitor SMD
0	5	0	0	0	0	**b**	Resistor SMD
0	0	5	0	0	0	**c**	Resistor nSMD
0	0	0	4	0	1	**d**	Schmitt Trigger nSMD
0	0	0	0	5	0	**e**	Transistor Pot nSMD
0	0	0	0	0	5	**f**	Transistor nSMD

4.7.3. Treinamento dos sistemas

Para que o sistema seja apto a avaliar os componentes da maneira esperada, é necessário treiná-lo utilizando as técnicas já descritas neste trabalho. Para isso, é preciso que todas as informações necessárias referente aos componentes *gold* e imagens de fundo de placa sejam extraídas e disponibilizadas, desta forma, o processo utilizado, quando se refere as imagens de fundo de placa, é representado pela figura 4.22.

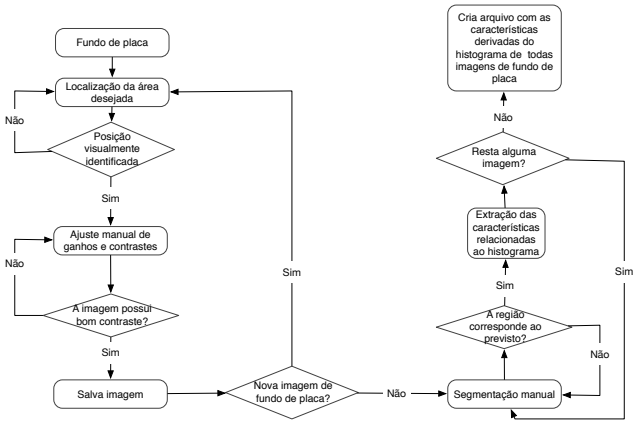


Figura 4.22: Processo de extração e disponibilização de características referente a imagens de fundo de placa.

O processo de localização da área desejada é feita movendo os eixos da S2iAOI de maneira manual. O ajuste de ganho e contraste é feito pelo software de gerenciamento da máquina, e ao fim deste processo a

imagem é salva do diretório do projeto vigente. Quando todas as imagens forem salvas, ocorre a segmentação manual da imagem, realizado pelo software de inspeção proposto neste trabalho, para selecionar o *frame*. A partir do quadro criado, as características derivadas do histograma são salvas em um arquivo, e este é disponibilizado para treinar classificador KNN.

Para as imagens *gold* a figura 4.23 representa o processo extração de características e treinamento dos sistemas de avaliação.

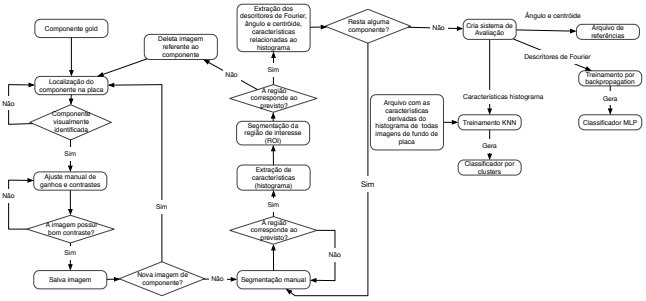


Figura 4.23: Processo de extração de características referente a imagens *gold* e treinamento dos sistemas de avaliação.

O processo se repete para localizar os componentes e salvá-los no diretório do projeto. O quadro resultante da segmentação manual é utilizado para extrair características derivadas do histograma, além de servir como entrada para o algoritmo de ROI. Caso a região apresentada pela segmentação realizada pelo algoritmo de ROI corresponda a esperada, as demais características são extraídas, e caso contrário, deleta-se a imagem do componente e o usuário deve gerar uma nova foto. Quando todos os componentes *gold* estiverem devidamente cadastrados, cria-se o sistema de avaliação, que é composto pelo classificador por clusters (que envolve imagens *gold* e imagens de fundo de placa), cujo objetivo é avaliar a existência ou ausência de componente, classificador MLP (utilizado para avaliar o tipo de componente), e um arquivo de referência utilizado para comparar ângulo de rotação e deslocamento entre componentes.

4.7.4. Avaliação de componente em teste

A avaliação dos componentes em teste segue a ordem de aquisição das imagens *gold*, logo, o controle da quantidade de componentes a serem

inspecionados é feito pela quantidade de componentes *gold* cadastrados. A figura 4.24 mostra o processo de avaliação de componentes em teste.

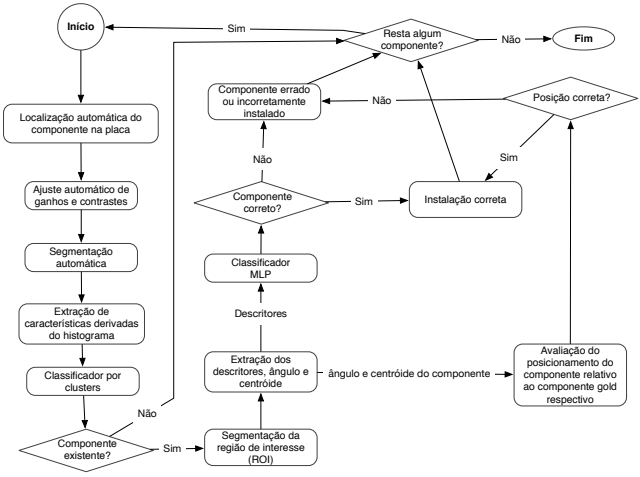


Figura 4.24: Processo de avaliação de componentes em teste.

A localização dos componentes é feita utilizando os mesmos valores cartesianos da aquisição do componente *gold* respectivo, assim como o ajuste de ganho e contraste, e segmentação automática, mantendo a métrica do quadro recortado. Ao extrair as características derivadas do histograma, ocorre a avaliação da existência ou ausência de componente (e não há comparação direta com o componente *gold* relativo), e caso não haja componente (onde deveria haver) se reinicia o processo de avaliação para o próximo componente a ser inspecionado. Caso haja algum componente, segue o processo de inspeção, onde a partir do quadro recortado, o algoritmo de ROI é performado, para que se obtenha as características relacionadas ao posicionamento e contorno do objeto. Para avaliar o componente relativo ao tipo, o processo segue pelo classificador MLP, resultando em componente correto ou errado quando comparado ao componente *gold*. No caso de avaliação do posicionamento, é comparado, com o componente *gold* respectivo, o ângulo e a centroide da maior elipse possível que se encaixa dentro do contorno gerado pela ROI, mantendo um taxa de tolerância de 5%. Ao fim do processo da avaliação de cada componente, as informações relativas à inspeção são salvas em uma arquivo, e este é disponibilizado ao agente, para que possa ser feita a avaliação de desempenho da produção.

4.7.5. Método de interação S2iAOI-agente

Para que o agente possa monitorar o processo de inspeção de PCI, é necessário que o sistema de inspeção se encontre pronto, ou seja, a máquina esteja calibrada, as imagens adquiridas e os sistemas de avaliação treinados. Com isso, o objetivo do agente consiste em calibrar a máquina, adquirir as imagens, treinar os sistemas de avaliação, inspecionar PCIs e monitorar esta inspeção, como é apresentado pelo diagrama de objetivos do agente, este representado pelo figura 4.25.

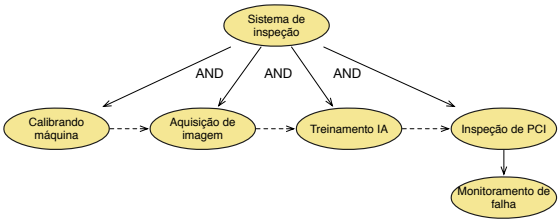


Figura 4.25: Diagrama de objetivos do agente de inspeção.

Desta maneira, a partir de informações provenientes da máquina S2iAOI, o agente pode tomar ação de interromper ou reiniciar o processo de inspeção, cuja decisão é tomada a partir da troca de mensagens com outros agentes, utilizando como parâmetros de decisão informações (percepções) referente ao status das etapas de treinamento, aquisição, calibração e inspeção, como é apresentado no diagrama de visão geral do sistema, pela figura 4.26.

As ações e percepções de interação entre o agente e artefato são explícitas no diagrama de visão geral do ambiente, apresentado pela figura 4.27.

O artefato de inspeção obtém informações da máquina S2iAOI por meio da operação *status*, cujo modo de representação destas informações é feito através de um conjunto de variáveis booleanas. Os parâmetros (também booleanos) utilizados para obter o status da máquina são constituídos pelos status de treinamento, aquisição, calibração e inspeção, desta forma, as propriedades observáveis são derivações lógicas dos parâmetros. A tabela 4.6 apresenta a refinação do artefato Inspeção.

A interação entre a máquina S2iAOI e o agente que a representa acontece através de um protocolo UDP (*User Datagram Protocol*), que estabelece a comunicação entre a máquina e um artefato de inspeção, responsável por disponibilizar as informações ao agente. A interação

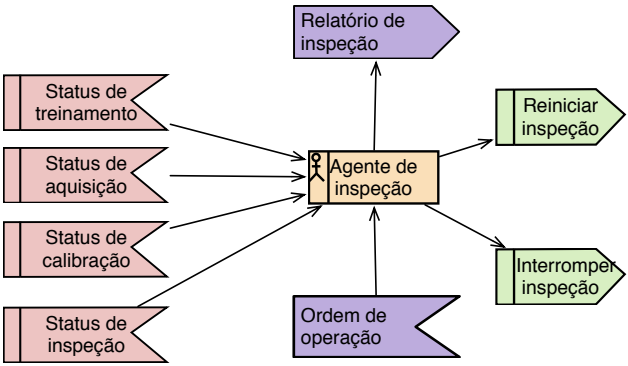


Figura 4.26: Diagrama de visão geral do sistema.

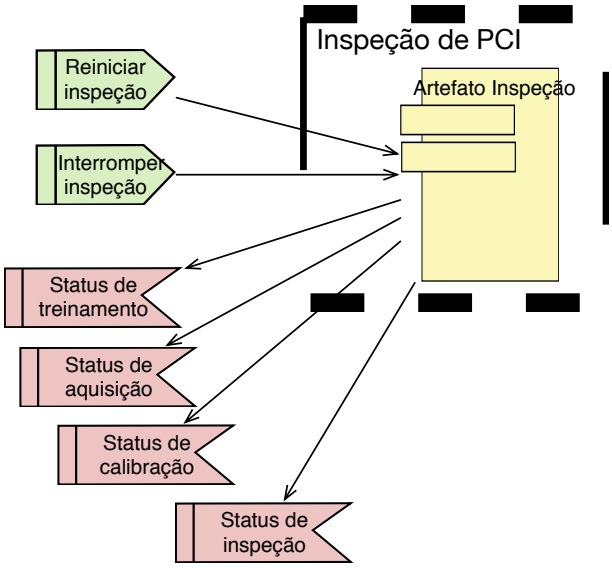


Figura 4.27: Diagrama de visão geral do ambiente.

entre agentes é feita através da troca de mensagens, esta abordada no trabalho de [ROLOFF, 2009]. O servidor UDP é localizado na S2iAOI, enquanto o cliente é situado no artefato, pois este tem autonomia para realizar consultas e chamar funções. Desta forma, há um canal de

Tabela 4.6: Descrição do artefato Inspeção.

Artefato Inspeção	
Descrição	Este artefato é utilizado para obter informações da máquina S2iAOI e servir como repositório de consulta para o agente.
Operações	status(boolean true/false)
Parâmetros	Boolean status_treinamento, status_aquisicao, status_calibracao, status_inspecao
Propriedades Observáveis	incmsg: mensagem entrando; training: sistema em treinamento; systrained: sistema treinado; getgold: adquirindo imagens de referência; goldok: imagens de referência adquiridas; calibrating: sistema calibrando; sysrdy2insp: sistema pronto para inspeção.
Eventos Observáveis	

comunicação, via socket, sempre aberto entre a máquina e o artefato de inspeção, e este é atualizado periodicamente.

A função do servidor situado na máquina é fornecer informações referente ao status da máquina e do sistema de inspeção, assim como relatórios e processo corrente da máquina. Após efetivar a conexão cliente-servidor, mensagens codificadas são trocadas utilizando *datagrams*, como mostra a figura 4.28.

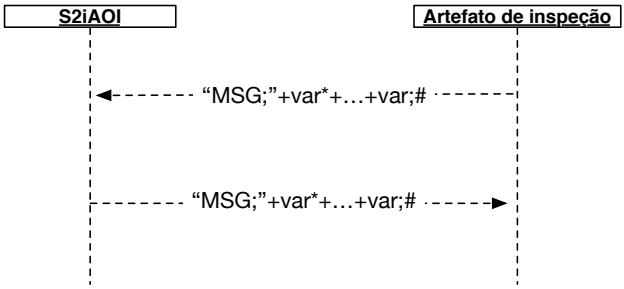


Figura 4.28: Processo de avaliação de componentes em teste.

As mensagens contém o identificador de início ("MSG;"), variáveis separadas por ";"e um identificador de fim de mensagem (";#"), para que o sistema ignore qualquer lixo anexado à mensagem durante a troca. Estas variáveis espelham medidores utilizados no software de inspeção, e representam informações de interesse do agente. A combinação destas variáveis geram as informações requeridas pelo agente, que são representados pela figura 4.29.

As variáveis existentes no sistema são criadas na classe "Co-

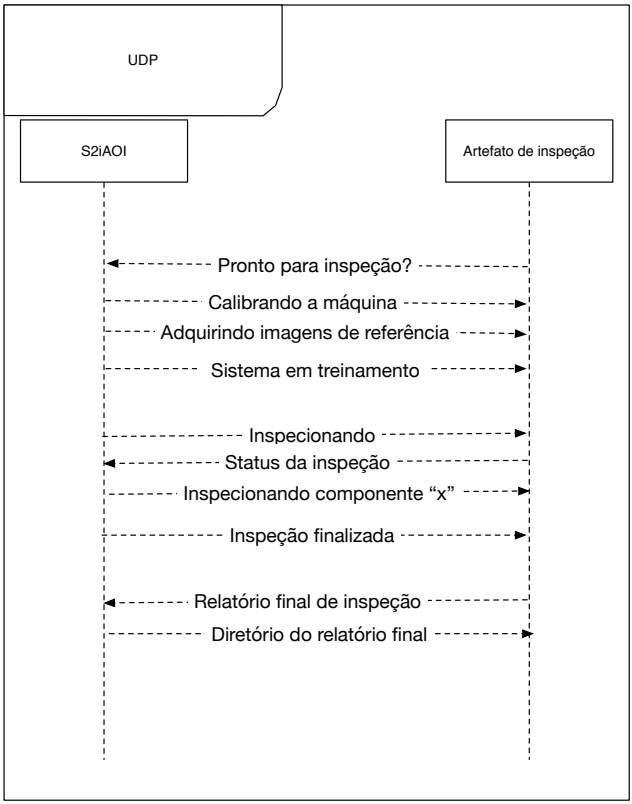


Figura 4.29: Troca de mensagens entre agente e S2iAOI.

muInfo", e são as seguintes:

- incmsg: mensagem entrando;
- training: sistema em treinamento;
- systrained: sistema treinado;
- getgold: adquirindo imagens de referência;
- goldok: imagens de referência adquiridas;
- calibrating: sistema calibrando;
- sysrdy2insp: sistema pronto para inspeção.

Desta forma, quando houver a necessidade de adicionar novos medidores, basta criar variáveis na classe ComuInfo e associá-las as variáveis devidas no sistema de inspeção.

O cliente UDP localizado no artefato é representado pela classe

Communication, e possui interação com o arquitetura do agente, como mostra a figura 4.30a. E esta interação fica mais evidente quando visto o diagrama de classes que representa a integração jason-java, apresentado pela figura 4.30b.

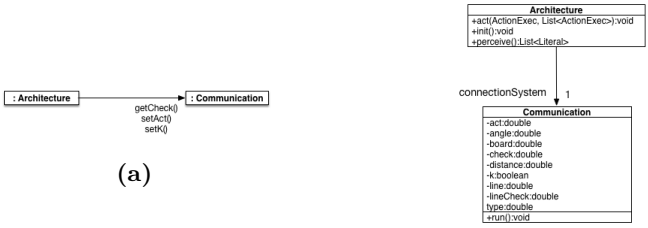


Figura 4.30: (a) Diagrama de interação e (b) diagrama de classes do protocolo de comunicação do artefato.

A classe *Communication* representa o cliente UDP e faz a troca de mensagens entre o artefato e a máquina S2iAOI, e as variáveis utilizadas na troca de mensagem são provenientes da classe *Architecture*, utilizando o sistema de *setter* e *getter* para a troca de informação entre classes. Por sua vez, a classe de arquitetura é composta por duas funções, *perceive* e *act*, que fazem a interface entre as variáveis nas linguagens java-jason. A função *perceive* tem como objetivo receber informações provenientes da máquina S2iAOI, enquanto a *act* envia informações. A taxa de atualização das informações é de 2 segundos, pois não há a necessidade de uma maior frequência de atualização, visto que o tempo de inspeção somado ao tempo de movimentação do eixo da máquina S2iAOI é de aproximadamente 5 segundos.

Capítulo 5

AVALIAÇÃO DOS RESULTADOS

Para validar o sistema de inspeção de componentes do tipo SMD proposto neste trabalho, foram realizados vários experimentos com componentes SMD, envolvendo defeitos encontrados em uma PCI e defeitos gerados via computação gráfica. A PCI utilizada como base para as avaliações é apresentada pela figura 5.1, e as imagens utilizadas nos experimentos foram adquiridas utilizando a máquina S2iAOI.

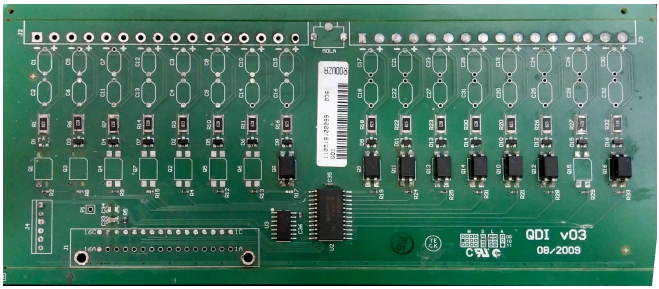


Figura 5.1: PCI utilizada no teste.

A aquisição de imagens segue o padrão criado por [MELO, 2013], ou seja, cria-se um projeto de inspeção e as imagens adquiridas são salvas em escala de cinza. Os seguintes componentes SMD, provenientes da PCI utilizada como base, foram utilizados: capacitor, resistor tipo 1, transistor de potência, transistor, schmitt trigger, resistor tipo 2 e imagem de fundo de placa. Estes componentes são expostos pela figura 5.2.

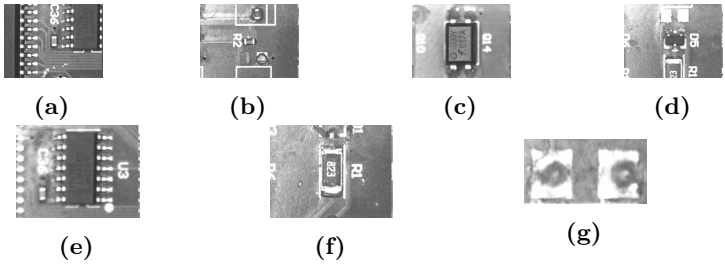


Figura 5.2: (a) Capacitor SMD, (b) resistor tipo 1 SMD, (c) transistor de potência SMD, (d) transistor SMD, (e) schmitt trigger SMD, (f) resistor tipo 2 SMD e (g) imagem de fundo de placa.

Dentre os componentes utilizados para a realização dos testes, alguns modelos apresentavam exemplares com defeitos de instalação, enquanto para os modelos que não haviam defeitos, estes foram gerados utilizando recursos de computação gráfica. Como abordado pelo capítulo 4.4, a inspeção de componentes SMD proposta neste trabalho consiste em verificar a existência, posicionamento e tipo do componente, com isso a figura 5.3 apresenta imagens, utilizadas em testes, que representam os defeitos de deslocamento, rotação e ausência de componente.

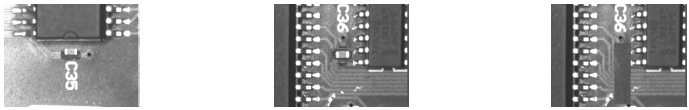
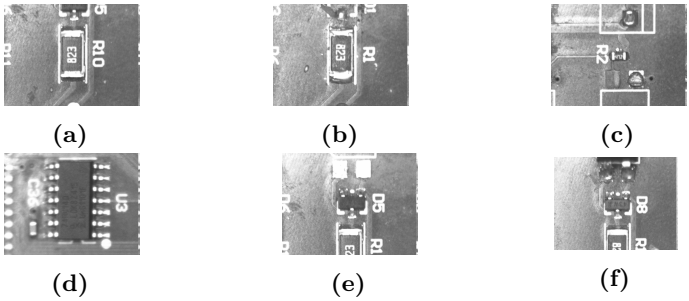


Figura 5.3: (a) Capacitor SMD deslocado, (b) capacitor SMD rotacionado e (c) ausência de capacitor SMD mantendo o padrão de fundo de placa.

Além das imagens com defeitos abordados por estes projeto, foram utilizadas imagens com diferentes níveis de ganho e contraste, presença de manchas ou outros componentes e falhas da aplicação de solda, como mostra a figura 5.4.

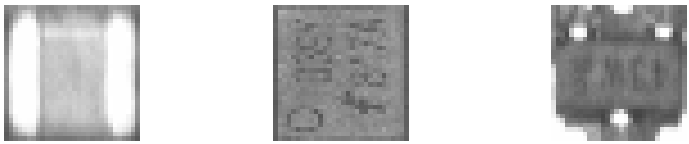


(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figura 5.4: (a) Resistor tipo 2, (b) resistor tipo 2 com falha na solda, (c) resistor tipo 1 com mancha, (d) imagem com schmitt trigger e capacitor, (e) imagem de transistor com ganho e contraste configuração 1 e (f) imagem de transistor com ganho e contraste configuração 2.

A primeira etapa consiste em avaliar o processo de obtenção da ROI, e para isso foram experimentados os 6 componentes da figura 5.3 submetidos a diferentes configurações de ganho e contraste. Para o capacitor, transistor de potência e o transistor, independente dos valores de configuração utilizados obteve-se a mesma região de interesse, representada pela figura 5.5.



(a) **(b)** **(c)**

Figura 5.5: Região de interesse do (a) capacitor, (b) transistor de potência e (c) transistor.

Para o schmitt trigger e os resistores houveram dois padrões apresentados como resultado do algoritmo de ROI, dependendo do nível de brilho utilizado para a obtenção da imagem, como é apresentado pela figura 5.6.

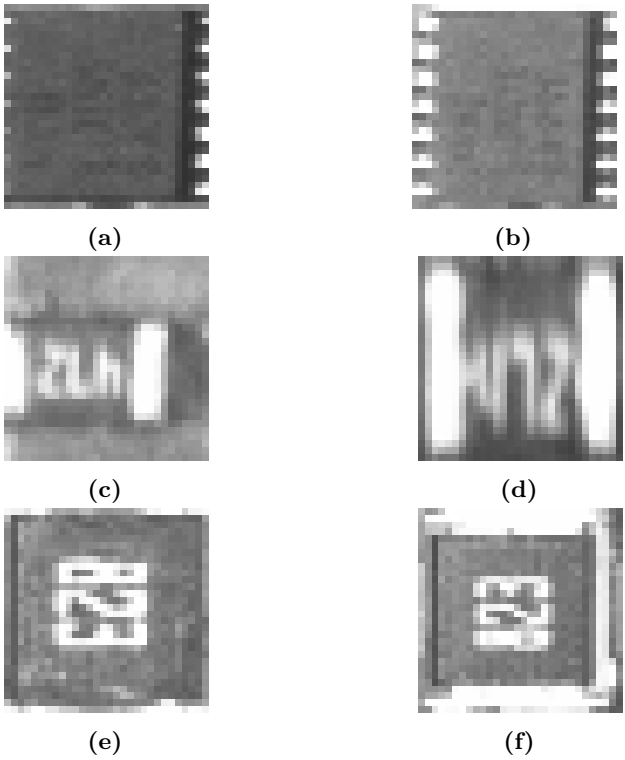


Figura 5.6: Região de interesse do (a) schmitt trigger com baixo brilho, (b) schmitt trigger com alto brilho, (c) resistor tipo 1 com alto brilho, (d) resistor tipo 1 com baixo brilho, (e) resistor tipo 2 com alto brilho e (f) resistor tipo 2 com baixo brilho.

A possibilidade de existir dois padrões diferentes, para um mesmo componente, gerados pela ROI não é indicado para o sistema de classificação proposto neste trabalho. Para 45 imagens obtidas, envolvendo os três tipos de componentes mostrados pela figura 5.6, houveram 1 caso de ROI não ideal para o resistor tipo 1 (figura 5.6c), 1 caso para o resistor tipo 2 (figura 5.6f) e 2 casos para o schmitt trigger (figura 5.6b). Quando a ROI não segue o padrão esperado, existe a possibilidade de se obter uma nova imagem, como é visto nos capítulos 4.7.3 e 4.7.4. A avaliação de desempenho sobre o método de pré-processamento e do algoritmo de ROI é apresentado pela tabela 5.1.

Tabela 5.1: Desempenho do algoritmo de ROI

	Baixo brilho Qtd/Erros	**Médio brilho** Qtd/Erros	**Alto brilho** Qtd/Erros
Capacitor	10/0	10/0	10/0
Resistor tipo 1	10/1	10/0	10/0
Resistor tipo 2	10/1	10/0	10/0
Schmitt Trigger	10/0	10/0	10/0
Transistor	10/0	10/0	10/0
Transistor de potência	10/0	10/0	10/0

Sendo assim, o sistema proposto para a extração da ROI é viável para aplicação proposta, pois para 150 componentes testados, apenas 3 componentes apresentaram falhas, onde todas ocorreram devido ao baixo valor de brilho utilizado, logo, poucas correções são necessárias para esta primeira etapa. O processo de extração de características não apresentou nenhum comportamento inesperado, pois este depende apenas do sucesso dos processos de seleção de quadro e reconhecimento da ROI.

A avaliação de componentes diz respeito apenas a ausência ou existência, translação, rotação e corretividade de tipo, não dando relevância para falhas em solda, trilhas, placa e componentes com erro de fabricação. Os defeitos de instalação reais foram produzidos fisicamente nas placas, enquanto os defeitos virtuais foram criados editando fotos obtidas de componentes, visando simular problemas como deslocamento, rotação e ausência de componentes. Utilizando o modelo de PCI apresentado pela figura 5.1, foram criados 5 casos de teste, o qual cada caso de teste representa uma inspeção.

- **Teste 1** - imagens gold: 5 diferentes exemplares de cada tipo de componente, constituído por: capacitor, resistor tipo 1, resistor tipo 2, transistor de potência, transistor e schmitt trigger, todos instalados devidamente.
  Fundo de placa: para cada imagem gold, uma imagem de fundo de placa.
  Imagens teste: mesma quantidade e modelos utilizados pelas imagens gold, onde para cada tipo de componente haviam 1

exemplar correto, 1 exemplar errado, 1 ausência de componente, 1 rotacionado, 1 transladado.

- **Teste 2** - imagens gold: 10 exemplares de cada tipo de componente, havendo a repetição de componentes mas com imagens adquiridas utilizando diferentes níveis de brilho, constituído por: capacitor, resistor tipo 1, resistor tipo 2, transistor de potência, transistor e schmitt trigger
  Fundo de placa: para cada imagem gold, uma imagem de fundo de placa.
  Imagens teste: mesma quantidade e modelos utilizados pelas imagens gold, onde para cada tipo de componente haviam 2 exemplares corretos, 2 exemplares errados, 2 ausências de componente, 2 rotacionados e 2 transladados.
- **Teste 3** - imagens gold: 5 exemplares de capacitores, não havendo a repetição de componentes.
  Fundo de placa: para cada imagem gold, uma imagem de fundo de placa.
  Imagens teste: mesma quantidade e modelos utilizados pelas imagens gold, onde para cada tipo de componente haviam 1 exemplar correto, 1 exemplar errado, 1 ausência de componente, 1 rotacionado, 1 transladado.
- **Teste 4** - imagens gold: 10 exemplares de capacitores, havendo a repetição de componentes mas com imagens adquiridas utilizando diferentes níveis de brilho.
  Fundo de placa: para cada imagem gold, uma imagem de fundo de placa.
  Imagens teste: mesma quantidade e modelos utilizados pelas imagens gold, onde para cada tipo de componente haviam 2 exemplares corretos, 2 exemplares errados, 2 ausências de componente, 2 rotacionados e 2 transladados.
- **Teste 5** - imagens gold: 10 exemplares de capacitores e 10 de transistores, havendo a repetição de componentes mas com imagens adquiridas utilizando diferentes níveis de brilho.
  Fundo de placa: para cada imagem gold, uma imagem de fundo de placa.
  Imagens teste: mesma quantidade e modelos utilizados pelas imagens gold, onde para cada tipo de componente haviam 2 exemplares corretos, 2 exemplares errados, 2 ausências de componente, 2 rotacionados e 2 transladados.

Para todos os casos de teste, foram utilizadas as mesmas configurações de classificadores apresentados pelos capítulos 4.7.1 e 4.7.2,

e para caso de teste foram feitos 10 ensaios. A tabela 5.2 apresenta um resumo contendo os resultados obtidos pelos casos de teste, cujo enfoque é dado na relação entre defeitos existentes ("ex") e encontrados ("en").

Tabela 5.2: Relação de desempenho entre defeitos existentes e encontrados obtidos nos casos de teste.

	Ausentes		**Rotacionados**		**Deslocados**		**Errados**		**Corretos**		**Total**	
Teste	ex	en	ex	en	ex	en	ex	en	ex	en	ex	en
1	6	6	6	5	6	5	6	5	6	5	30	26
2	12	12	12	11	12	11	12	10	12	10	60	54
3	1	1	1	1	1	1	1	0	1	1	5	4
4	2	2	2	2	2	2	2	0	2	2	10	8
5	4	4	4	4	4	4	4	4	4	4	20	20
Total	25	25	25	23	25	23	25	20	25	22	125	112
%	**100%**		**92%**		**92%**		**80%**		**88%**		**89.6%**	

A maioria dos testes realizados apresentaram um aproveitamento total de 100%, e este é totalmente dependente do sucesso das etapas de pré-processamento e extração de características, como mostra a tabela 5.3, cujo enfoque é dado para a relação entre componentes adquiridos para inspeção ("comp") e componentes que resultaram na ROI ideal ("suces"), isto é, obteve-se sucesso nas duas primeiras etapas do processo.

Tabela 5.3: Relação de dependência entre as etapas de pré-processamento, extração de características e avaliação.

	Pré-processamento		**Extração de características**		**Avaliação**	
Teste 1	comp	suces	comp	suces	comp	suces
Ensaio 1	30	30	30	30	**30**	**30**
Ensaio 2	30	30	30	30	**30**	**30**
Ensaio 3	30	28	30	28	**30**	**28**
Ensaio 4	30	30	30	30	**30**	**30**
Ensaio 6	30	30	30	30	**30**	**30**
Ensaio 7	30	30	30	30	**30**	**30**
Ensaio 8	30	26	30	26	**30**	**26**
Ensaio 9	30	30	30	30	**30**	**30**
Ensaio 10	30	30	30	30	**30**	**30**

Como já mencionado anteriormente, a região de interesse resultante da etapa de pré-processamento é utilizada como base para a extração de características, logo quando há alguma falha nesta etapa, esta é propagada para as etapas conseguintes, como mostra a tabela 5.3. O comportamento de propagação de erro apresentado pelo teste 1 se reflete para o teste 2, e para ambos casos, os problemas encontrados

pela avaliação incorreta sobre o tipo de componente são frutos de uma extração de ROI má sucedida, ocorrida devido ao excesso de brilho utilizado ao adquirir a imagem. Nos erros de rotação e deslocamento, como a ROI obtida pelos componentes teste e gold são semelhantes mas não idênticas, quando um componente teste está ligeiramente rotacionado/transladado, sua ROI é muito parecida à de componente gold não rotacionado/transladado, e como a margem de 5% de erro é utilizada, a avaliação pode não identificar a rotação/translação. Os testes 3 e 4 utilizam apenas um tipo de componente para treinamento e teste, desta maneira os erros encontrados traduzem um comportamento esperado para o método de classificação proposto, pois quando não há mais de um tipo de componente cadastrado, a RNA não é capaz de reconhecer componentes diferentes dos treinados. O teste 5 não apresentou erros de avaliação.

Dentre os requisitos apresentados nas seções 4.1.1 e 4.1.2, conclui-se que o escopo do sistema (requisito funcional 1) foi atingido, pois o método proposto, desenvolvido na forma de plugin, realiza a inspeção, de maneira individual, de componentes do tipo SMD posicionados em uma PCI. Os tipos de defeitos descritos no requisito funcional 2 foram devidamente identificados. A comunicação entre a máquina S2iAOI e o agente (requisito funcional 3) foi estabelecida com sucesso, mantendo um canal exclusivo e ininterrupto de comunicação. Existe compatibilidade entre o software de inspeção proposto e o software de gerência da máquina S2iAOI (requisito funcional 4), pois ambos foram desenvolvidos na linguagem Java, contudo não foi validado através de experimentos.

O tempo de setup (requisito não-funcional 1) e de inspeção (requisito não funcional 2) foram menores que o tempo geralmente tomando em uma produção em larga escala. O tempo de inspeção de um projeto é composto pelo tempo de calibração de máquina, aquisição de imagens gold e de fundo de tela, treinamento e inspeção de componentes teste. O tempo de calibração (setup) depende da velocidade da movimentação dos eixos da máquina e do algoritmo de *fiducial finder*, ambos propostos pelo trabalho de [MELO, 2013]. A aquisição de imagem depende da velocidade de movimentação dos eixos, selecionamento dos ganhos de contraste e brilho, recorte do quadro e extração de características, onde o tempo dos dois últimos itens citados não passa de 1 segundo (contando a partir do momento que o operador seleciona o quadro). O tempo gasto para treinar o classificador KNN com 100 imagens, utilizando o vetor proposto por este trabalho para representar cada imagem, demorou menos de 1 segundo. O classificador MLP gastou aproximadamente 30

segundos para treinar quando foram utilizadas 100 imagens, enquanto para as 60 imagens utilizadas no teste 2, o tempo foi de 5.14 segundos. O tempo de inspeção é computado pela calibração da máquina, aquisição de imagens, extração de características e avaliação do componente, onde os dois últimos itens não ultrapassa 2 segundos. Com isso, o maior tempo de inspeção constatado ocorreu durante o caso de teste 2, gastando 35 minutos para os processos que envolvem as imagens gold e de fundo de tela, e 14 minutos para o processo de inspeção, considerando que o tempo de deslocamento do eixo é muito maior que o tempo de aquisição de imagem e avaliação.

O método de inspeção proposto apresentou confiabilidade suficiente para solucionar o problema em questão, de modo que os erros encontrados ocorreram devido a aquisição de imagens não ideais, havendo a possibilidade de readquirir imagens caso as características extraídas não correspondam ao esperado. Desta forma, cabe ao usuário adquirir imagens com bons níveis de ganho e contraste.

Alguns problemas relacionados a rotação, deslocamento e iluminação não ocorreram durante os testes, mas vale ressaltá-los. Os erros de rotação e deslocamentos podem também ocorrer devido a: ou pela extração da ROI má sucedida ou um deslocamento/posicionamento indevido na placa durante inspeção. Quando a ROI obtida pelo componente teste não segue o padrão obtido pelo componente gold respectivo, pode gerar um falso negativo, onde o componente em teste pode estar corretamente instalado, porém devido a uma falha na aquisição de imagem, diferentes níveis de brilho ou contraste são utilizados. Como mencionado no capítulo 4.7.4, a inspeção de componentes teste segue a mesma ordem de cadastro de componentes gold, pois são utilizadas as mesmas coordenadas cartesianas para identificar a posição do componente, contudo, quando há uma movimentação indevida durante a inspeção, não há um recalibramento da máquina, e as posições relativas às marcas fiduciais utilizadas no cadastramento de componentes gold não traduzem a mesma posição entre componentes gold e teste. Os erros de rotação e deslocamento também podem ocorrer devido a um posicionamento enviesado da placa teste na máquina S2iAOI, pois a posição relativa é obtida pelo algoritmo de *Fiducial Finder*, ou procurador fiducial, desenvolvido por [MELO, 2013], e este utiliza apenas 2 marcas fiduciais, não sendo capaz criar um posicionamento relativo para placas posicionadas de forma angulada. Ao criar um projeto, é importante utilizar o mesmo modo de iluminação para a aquisição de todas as imagens, pois todos os algoritmo de pré-processamento utilizados não são invariantes à mudanças de iluminação, como é mostrado no capítulo

de fundamentação.

Capítulo 6

CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvido um software de inspeção por processamento de imagens com o objetivo de inspecionar, de maneira automática, componentes SMD, sem qualquer marcação, instalados em uma PCI durante uma PPS, e um protocolo de comunicação para viabilizar a interação entre a máquina de inspeção S2iAOI e o agente que a representa.

A abordagem utilizada na inspeção composta pelo processamento de imagem morfológica, segmentação de imagem, técnicas de detecção de borda, descrição de objetos por meio de descritores e classificação de padrões via técnicas de inteligência artificial mostrou-se eficaz para a inspeção óptica automática de componentes do tipo SMD durante uma PPS.

O software desenvolvido é apto para ser utilizado em ambiente, dentro das restrições já mencionadas por este projeto, como mostram os testes realizados. Os métodos de classificação mostraram-se eficazes e pertinentes à aplicação proposta, pois não há a necessidade de utilizar maus exemplos no treinamento, além da validação cruzada comprovar a capacidade de adaptação destes métodos para diferentes aplicações.

O protocolo de comunicação agente-máquina mostrou-se eficiente para o caso presente, pois este mantém um canal aberto para a troca de mensagens exclusivo para os comunicantes em questão.

O método de inspeção proposto neste trabalho gerou uma publicação no evento internacional ISIE 2015 (*International Symposium on Industrial Electronics*) [MELLO; STEMMER, 2015].

Ressalta-se na abordagem proposta de inspeção a não necessidade da utilização de maus-exemplos para o treinamento, evitando o desperdício de recursos. Outro ponto positivo é a não necessidade de especificar o tipo de defeito a ser analisado, pois o método proposto

avalia o componente para todos os problemas descritos neste projeto, sem elevados custos computacionais.

A limitação presente no sistema de inspeção proposto existe quando há componentes com dimensões muito semelhantes, gerando características parecidas para diferentes tipos de componentes, que por sua vez serão inspecionados de maneira incorreta. Com o uso de técnicas adicionais, como análise de serigrafia, como proposto por [SZYMANSKI, 2014], estas limitações podem ser diminuídas.

Ressalta-se que apesar o projeto visar a inspeção de componentes durante uma produção em PPS, as técnicas de classificação e o método de integração agente-máquina podem ser utilizados em contextos variados.

De modo geral, a abordagem utilizada neste projeto obteve boa taxa de acertos, mostrando ser robusta o suficiente para realizar inspeções de componentes do tipo SMD, sem serigrafia, durante uma PPS.

6.1. TRABALHOS FUTUROS

Os aspectos observados que podem ser melhorados ou aprofundados, com o objetivo de expandir este trabalho são:

- a abordagem de inspeção utilizada neste projeto é baseada no formato do componente e não considera marcação, serigrafia ou textura presente na superfície dos componentes. Desta maneira, sugere-se a integração do método proposto com diferentes abordagens, como com o trabalho desenvolvido por [SZYMANSKI, 2014];
- visando aumentar a robustez nas etapas de pré-processamento e extração de características, indica-se utilizar imagens coloridas, pois em cada canal de cor há uma matriz que representa a imagem, como é proposto em [WU et al., 2009];
- não é abordado neste projeto a utilização de arquivos CAD referente a componentes ou PCI. No entanto, sugere-se um estudo para viabilizar o uso de informações fornecidas por um arquivo CAD na etapa de treinamento;
- a inspeção de ângulo e posição do componente é feita de maneira simples, desta forma, propõe-se o estudo de um método mais robusto para realizar esta verificação;
- o algoritmo *fiducial finder*, responsável por identificar a posição relativa da PCI, não considera a rotação da placa. Desta maneira, uma melhoria é o desenvolvimento de um algoritmo que compense uma eventual rotação na posição relativa da PCI;

- o agente que representa a máquina de inspeção S2iAOI possui uma programação simples, onde apenas consultas são realizadas. Cabe a uma próxima etapa, desenvolver a interação entre o agente e a máquina de inspeção, para que o agente controle a máquina por completo.

Referências

ACCIANI, G.; BRUNETTI, G.; FORNARELLI, G. Application of Neural Networks in Optical Inspection and Classification of Solder Joints in Surface Mount Technology. **IEEE Transactions on Industrial Informatics**, v. 2, n. 3, p. 200–209, ago. 2006. ISSN 1551-3203.

AHA, D. Editorial. **Artificial Intelligence Review**, Kluwer Academic Publishers, v. 11, n. 1-5, p. 7–10, 1997. ISSN 0269-2821.

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. **Machine Learning**, v. 6, n. 1, p. 37–66, jan. 1991. ISSN 0885-6125.

BATISTA, G. E. A. P. A.; SILVA, D. F. How k-nearest neighbor parameters affect its performance. In: **Argentine Symposium on Artificial Intelligence**. [S.l.: s.n.], 2009. p. 1–12.

BAY, H.; ESS, A.; TUYTELAARS, T.; Van Gool, L. Speeded-Up Robust Features (SURF). **Computer Vision and Image Understanding**, v. 110, n. 3, p. 346–359, jun. 2008. ISSN 10773142.

BERGHOLM, F. Edge Focusing. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-9, n. 6, p. 726–741, nov. 1987. ISSN 0162-8828.

BEUCHER, S.; LANTUEJOUL, C. Use of Watershed in contour detection. **International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation**, p. 2.1–2.12, 1979.

BEUCHER, S.; MEYER, F. **Chapter 12 The Morphological**

Approach to Segmentation : The Watershed Transformation. [S.l.]: E. R. Dougherty, 1993. 433 – 481 p.

Bo Peng; Tianrui Li. A Probabilistic Measure for Quantitative Evaluation of Image Segmentation. **IEEE Signal Processing Letters**, v. 20, n. 7, p. 689–692, jul. 2013. ISSN 1070-9908.

BOISSIER, O.; BORDINI, R. H.; HUBNER, J. F.; RICCI, A.; SANTI, A. Multi-agent oriented programming with jacamo. **Sci. Comput. Program.**, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 78, n. 6, p. 747–761, jun. 2013. ISSN 0167-6423. Disponível em: <http://dx.doi.org/10.1016/j.scico.2011.10.004>.

BORDINI, R. H.; HUBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using Jason**. [S.l.]: John Wiley & Sons, 2007. ISBN 0470029005.

CANNY, J. A Computational Approach to Edge Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-8, n. 6, p. 679–698, nov. 1986. ISSN 0162-8828.

CHENG, Y. Mean shift, mode seeking, and clustering. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 17, n. 8, p. 790–799, 1995. ISSN 01628828.

COSTA, C. P. B. d. **Desenvolvimento de um Sistema de Diagnóstico de Defeitos na Montagem de PCI Baseado em Redes Bayesianas**. 102 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2012.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, jan. 1967. ISSN 0018-9448.

Dengsheng Zhang; Guojun Lu. Content-based shape retrieval using different shape descriptors: a comparative study. In: **IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.** [S.l.]: IEEE, 2001. p. 1139–1142. ISBN 0-7695-1198-8.

DORO, M. M. **Solução integrada para auxiliar na garantia da qualidade de produção em pequenos lotes**. 127 p. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2009.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. 2nd. ed. [S.l.]: Wiley-Interscience, 2000. ISBN 0471056693.

FORSYTH, D. A.; PONCE, J. **Computer Vision: A Modern Approach**. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130851981.

FRANK, U.; GIESE, H.; KLEIN, F.; OBERSCHELP, O.; SCHMIDT, A.; SCHULZ, B.; VöCKING, H.; WITTING, K. **Selbstoptimierende Systeme des Maschinenbaus - Definitionen und Konzepte**. [S.l.: s.n.], 2004. v. 155.

FUKUNAGA, K.; HOSTETLER, L. The estimation of the gradient of a density function, with applications in pattern recognition. **IEEE Transactions on Information Theory**, IEEE, v. 21, n. 1, p. 32–40, jan. 1975. ISSN 0018-9448.

GENESERETH, M. R.; KETCHPEL, S. P. Software agents. **Commun. ACM**, ACM, New York, NY, USA, v. 37, n. 7, p. 48–ff., jul. 1994. ISSN 0001-0782.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing (3rd Edition)**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. 976 p. ISBN 013168728X.

GRANLUND, G. H. Fourier Preprocessing for Hand Print Character Recognition. **IEEE Transactions on Computers**, C-21, n. 2, p. 195–201, fev. 1972. ISSN 0018-9340.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: An update. **SIGKDD Explor. Newsl.**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/1656274.1656278>.

Han-Jin Cho; Tae-Hyung Park. Template matching method for SMD inspection using discrete wavelet transform. In: **2008 SICE Annual Conference**. [S.l.]: IEEE, 2008. p. 3198–3201. ISBN 978-4-907764-30-2.

HARALICK, R. M.; STERNBERG, S. R.; ZHUANG, X. Image Analysis Using Mathematical Morphology. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-9, n. 4, p.

532–550, jul. 1987. ISSN 0162-8828.

HART, P. The condensed nearest neighbor rule (Corresp.). **IEEE Transactions on Information Theory**, v. 14, n. 3, p. 515–516, maio 1968. ISSN 0018-9448.

HAYKIN, S. **Neural Networks and Learning Machines**. 3rd. ed. [S.l.]: Prentice Hall, 2008. Hardcover. ISBN 0131471392.

HAZURAH, S.; PUTERA, I.; DZAFARUDDIN, S. F.; MOHAMAD, M.; MARA, U. T. MATLAB Based Defect Detection and Classification of Printed Circuit Board. **IEEE - Digital Information and Communication Technology and it's Applications (DICTAP)**, p. 115–119, 2012.

HEATH, M.; SARKAR, S.; SANOCKI, T.; BOWYER, K. A robust visual method for assessing the relative performance of edge-detection algorithms. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 19, n. 12, p. 1338–1359, dez. 1997. ISSN 0162-8828.

HITOMI, K. **Manufacturing Systems Engineering: A Unified Approach to Manufacturing Technology, Production Management, and Industrial Economics**. 2nd. ed. London: CRC Press, 1996. 536 p.

HUANG, S.-H.; PAN, Y.-C. Automated visual inspection in the semiconductor industry: A survey. **Computers in Industry**, v. 66, p. 1–10, jan. 2015. ISSN 01663615.

Indera Putera, S.; IBRAHIM, Z. Printed circuit board defect detection using mathematical morphology and MATLAB image processing tools. In: **2010 2nd International Conference on Education Technology and Computer**. [S.l.]: IEEE, 2010. v. 5, p. V5–359–V5–363. ISBN 978-1-4244-6367-1.

IPC, A. C. E. I. **North American Electronics Industry Had a Disappointing December**. 3000 E. Lakeside Drive, 105 N, Bannockburn, IL 60015 USA, 2014. Online; acessado 25 de Outubro de 2015.

IVERSON, L.; ZUCKER, S. Logical/linear operators for image curves. **IEEE Transactions on Pattern Analysis and Machine**

Intelligence, v. 17, n. 10, p. 982–996, 1995. ISSN 01628828.

JADHAV, S. a. **Setup approval and self starting schemes for short production runs**. 103 p. Dissertação (Mestrado) — University of Pune, 2005.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G. **Machine Vision**. 1st. ed. [S.l.]: McGraw-Hill, Inc., 1995. ISBN 0070320187.

KAEHLER, A.; BRADSKI, G. **Learning OpenCV**. 2nd. ed. 1005 Gravenstein Highway North, Sebastopol: O'Reilly Media, 2015. 575 p. ISBN 978-1-4493-3193-1.

KEARNS, M.; RON, D. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In: **Proceedings of the Tenth Annual Conference on Computational Learning Theory**. New York, NY, USA: ACM, 1997. (COLT '97), p. 152–162. ISBN 0-89791-891-6.

KOHAVI, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. **Proceedings of the 14th international joint conference on Artificial intelligence**, v. 2, p. 1137 – 1143, 1995.

KRIG, S. **Computer Vision Metrics: Survey, Taxonomy, and Analysis**. 1st. ed. Berkely, CA, USA: Apress, 2014. ISBN 1430259299, 9781430259299.

LALA, A.; JAIN, N. Image Segmentation: A Short Survey. In: **Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)**. [S.l.]: Institution of Engineering and Technology, 2013. p. 7.12–7.12. ISBN 978-1-84919-846-2.

LETA, F.; FELICIANO, F.; MARTINS, F. Computer vision system for printed circuit board inspection. **ABCM Symposium Series in Mechatronics**, v. 3, p. 623–632, 2008.

LIN, S.-c.; SU, C.-h. A Visual Inspection System for Surface Mounted Devices on Printed Circuit Board. In: **2006 IEEE Conference on Cybernetics and Intelligent Systems**. [S.l.]: IEEE, 2006. p. 1–4. ISBN 1-4244-0022-8.

LIN, S.-Y.; LAI, Y.-J.; CHANG, S. I. SHORT-RUN STATISTICAL PROCESS CONTROL: MULTICRITERIA PART FAMILY FORMATION. **Quality and Reliability Engineering International**, v. 13, n. 1, p. 9–24, jan 1997.

LOWE, D. Object recognition from local scale-invariant features. **Proceedings of the Seventh IEEE International Conference on Computer Vision**, Ieee, p. 1150–1157 vol.2, 1999.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 60, n. 2, p. 91–110, nov. 2004. ISSN 0920-5691.

LUO, B.; GAO, Y.; SUN, Z.; ZHAO, S. SMT Components Model Inspection Based on Characters Image Matching and Verification. **2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing**, Ieee, p. 1438–1441, ago. 2013.

LV, H.; WU, Z. A Survey of Computer Graphics and Graphics Image Processing Technology. In: **2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing**. [S.l.]: IEEE, 2011. p. 196–198. ISBN 978-1-4577-1130-5.

MARQUES, O. **Practical image and video processing using MATLAB**. 1st. ed. [S.l.]: Wiley/IEEE Press, 2011. 696 p.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biology**, Kluwer Academic Publishers, v. 5, n. 4, p. 115–133, dez. 1943. ISSN 0007-4985.

MELLO, A. R. d.; STEMMER, M. R. Inspecting surface mounted devices using k nearest neighbor and multilayer perceptron. **International Symposium on Industrial Electronics**, IEEE, 2015.

MELO, D. F. F. d. **Desenvolvimento de uma Máquina de Inspeção Automática de Placas de Circuito Impresso para Produção em Pequenas Séries**. 114 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2013.

MIKOLAJCZYK, K.; SCHMID, C. Indexing based on scale invariant interest points. In: **Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001**. [S.l.]: IEEE Comput. Soc, 2001. v. 1, p. 525–531. ISBN 0-7695-1143-0.

MOGANTI, M.; ERCAL, F.; DAGLI, C. H.; TSUNEKAWA, S. Automatic PCB Inspection Algorithms: A Survey. **Computer Vision and Image Understanding**, v. 63, n. 2, p. 287–313, mar. 1996. ISSN 10773142.

NALWA, V. S.; BINFORD, T. O. On Detecting Edges. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-8, n. 6, p. 699–714, nov. 1986. ISSN 0162-8828.

NETRAVALI, A. N.; HASKELL, B. G. **Digital Pictures: Representation and Compression**. [S.l.]: Perseus Publishing, 1988. ISBN 0306427915.

NIXON, M.; AGUADO, A. S. **Feature Extraction & Image Processing**. 2nd. ed. [S.l.]: Academic Press, 2008. ISBN 0123725380, 9780123725387.

O'CONCHUIR, D.; MCCURDY, J.; CASEY, V. Survey of non-destructive inspection methods for solder joint integrity. In: **Proceedings of the IEEE 1991 National Aerospace and Electronics Conference NAECON 1991**. [S.l.]: IEEE, 1991. p. 1268–1275. ISBN 0-7803-0085-8.

OTSU, N. A Tlreshold Selection Method from Gray-Level Histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, C, n. 1, p. 62–66, 1979.

PERNG, D.-B.; LIU, H.-W.; CHANG, C.-C. Automated SMD LED inspection using machine vision. **The International Journal of Advanced Manufacturing Technology**, v. 57, n. 9-12, p. 1065–1077, abr. 2011. ISSN 0268-3768.

PFEIFER, T.; SCHMITT, R.; PAVIM, A.; STEMMER, M.; ROLOFF, M.; SCHNEIDER, C.; DORO, M. Cognitive production metrology: A new concept for flexibly attending the inspection requirements of small series production. In: HINDUJA, S.; LI, L. (Ed.). **Proceedings of the 36th International MATADOR Conference**. [S.l.]: Springer

London, 2010. p. 359–362. ISBN 978-1-84996-431-9.

PLATT, J. **Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines**. [S.l.], 1998. 21 p.

RAUT, S. A.; RAGHUWANSHI, M.; DHARASKAR, R.; RAUT, A. Image Segmentation; A State-Of-Art Survey for Prediction. In: **2009 International Conference on Advanced Computer Control**. [S.l.]: IEEE, 2009. p. 420–424. ISBN 978-0-7695-3516-6.

RIBEIRO, J. C. d. C. e. S. **Uma Experiência de Agentificação aplicada a Software Educacional**. 197 p. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2003.

ROLOFF, M. L. **Uma nova Abordagem para a Implementação de um Sistema Multiagente para a Configuração e o Monitoramento da Produção de Pequenas Spéries**. 127 p. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, 2009.

ROTHWELL, C.; MUNDY, J.; HOFFMAN, W.; NGUYEN, V.-D. Driving vision by topology. In: **Proceedings of International Symposium on Computer Vision - ISCV**. [S.l.]: IEEE Comput. Soc. Press, 1995. p. 395–400. ISBN 0-8186-7190-4.

RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations**. Cambridge, MA, USA: MIT Press, 1986. ISBN 0-262-68053-X.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007. ISBN 0136042597, 9780136042594.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

SAVAGE, R. M.; PARK, H. S.; FAN, M. S. **Automated inspection of solder joints for surface mount technology**. NASA Goddard Space Flight Center; Greenbelt, MD, 1993. Online; accessed 15 January 2015.

SCHMITT, R.; PAVIM, A. Flexible optical metrology strategies for the control and quality assurance of small series production. **Proceedings of SPIE**, v. 7389, p. 738902–738902–12, 2009. ISSN 0277786X.

STEMMER, M. R.; COSTA, C. P. B. d.; VARGAS, J.; ROLOFF, M. L. Artificial intelligent systems for quality assurance in small series production. **Key Engineering Materials**, Trans Tech Publications, Switzerland, v. 613, p. 279–287, May 2014.

STOCKMAN, G.; SHAPIRO, L. G. **Computer Vision**. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130307963.

SUZUKI, S.; BE, K. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, v. 30, n. 1, p. 32–46, abr. 1985. ISSN 0734189X.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN 1848829345, 9781848829343.

SZYMANSKI, C. **Pesquisa e Desenvolvimento de Técnicas de Processamento Digital de Imagens para Garantia da Qualidade em Linhas de Produção de Placas de Circuito Impresso em Pequenas Séries**. 114 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2014.

TAN, H.; GELFAND, S.; DELP, E. A comparative cost function approach to edge detection. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, v. 19, n. 6, p. 1337–1349, 1989. ISSN 00189472.

TEOH, E.; MITAL, D.; LEE, B.; WEE, L. Automated visual inspection of surface mount PCBs. In: **[Proceedings] IECON '90: 16th Annual Conference of IEEE Industrial Electronics Society**. [S.l.]: IEEE, 1990. p. 576–580. ISBN 0-87942-600-4.

VIJAYARANI1, M. S.; MUTHULAKSHMI, M. M. Comparative Analysis of Bayes and Lazy Classification Algorithms. **International Journal of Advanced Research in Computer and Communication Engineering**, v. 2, n. 8, p. 3118–3124, August 2013. ISSN 2319-5940 / 2278-1021.

WANG, Y.; SUN, Y.; ZHANG, W. Automatic Inspection of Small

Component on Loaded PCB Based on Mean-Shift and Support Vector Machine. **2009 Fifth International Conference on Natural Computation**, Ieee, p. 195–198, 2009.

WEISS, G. (Ed.). **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence**. Cambridge, MA, USA: MIT Press, 1999. ISBN 0-262-23203-0.

WEISS, S. M.; KULIKOWSKI, C. A. **Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. ISBN 1-55860-065-5.

WIDROW, B.; HOFF, M. E. Adaptive Switching Circuits. In: **1960 IRE WESCON Convention Record, Part 4**. New York: IRE, 1960. p. 96–104.

WOOLDRIDGE, M. Agent-based software engineering. In: **EE Proceedings on Software Engineering**. [S.l.: s.n.], 1997. p. 26–37.

WOOLDRIDGE, M.; JENNINGS, N. R. Pitfalls of agent-oriented development. In: **Proceedings of the Second International Conference on Autonomous Agents**. New York, NY, USA: ACM, 1998. (AGENTS '98), p. 385–391. ISBN 0-89791-983-1.

WOOLRIDGE, M.; WOOLDRIDGE, M. J. **Introduction to Multiagent Systems**. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047149691X.

WU, H.; FENG, G.; LI, H.; ZENG, X. Automated visual inspection of surface mounted chip components. In: **2010 IEEE International Conference on Mechatronics and Automation**. [S.l.]: IEEE, 2010. p. 1789–1794. ISBN 978-1-4244-5140-1. ISSN 2152-7431.

WU, H.-H.; ZHANG, X.-M.; HONG, S.-L. A visual inspection system for surface mounted components based on color features. In: **2009 International Conference on Information and Automation**. [S.l.]: IEEE, 2009. p. 571–576. ISBN 978-1-4244-3607-1.

WU, W.-Y.; WANG, M.-J. J.; LIU, C.-M. Automated inspection of printed circuit boards through machine vision. **Computers in Industry**, v. 28, n. 2, p. 103–111, maio 1996. ISSN 01663615.

ZAHN, C. T.; ROSKIES, R. Z. Fourier Descriptors for Plane Closed Curves. **IEEE Transactions on Computers**, C-21, n. 3, p. 269–281, mar. 1972. ISSN 0018-9340.

ZHANG, D.; LU, G. Review of shape representation and description techniques. **Pattern Recognition**, v. 37, n. 1, p. 1–19, jan. 2004. ISSN 00313203.